aws

用户指南

Amazon Simple Storage Service

API 版本 2006-03-01

Amazon Simple Storage Service: 用户指南

Table of Contents

什么是 Amazon S3？

Amazon Simple Storage Service（Amazon S3）是一种对象存储服务，提供行业领先的可扩展性、数据可用性、安全性和性能。各种规模和行业的客户都可以使用 Amazon S3 存储和保护任意数量的数据，用于数据湖、网站、移动应用程序、备份和恢复、归档、企业应用程序、IoT 设备和大数据分析。Amazon S3 提供了管理功能，使您可以优化、组织和配置对数据的访问，以满足您的特定业务、组织和合规性要求。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅什么是 S3 Express One Zone？和目录桶。

主题

Amazon S3 的功能

存储类

Amazon S3 提供一系列适合不同使用案例的存储类。例如，您可以将任务关键型生产数据存储在 S3 Standard 或 S3 Express One Zone 中以便频繁访问，将不经常访问的数据存储在 S3 Standard-IA 或 S3 One Zone-IA 中，并在 S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、和 S3 Glacier Deep Archive 中以极低的成本归档数据。

Amazon S3 Express One Zone 是高性能的单区 Amazon S3 存储类，专门用于为延迟要求极高的应用程序提供稳定的毫秒级数据访问。S3 Express One Zone 是目前具有极低延迟的云对象存储类，相比

S3 Standard，其数据访问速度要快 10 倍，且请求成本低 50%。S3 Express One Zone 是第一种可以在其中选择单个可用区的 S3 存储类，您可以选择将您的对象存储与计算资源联合托管在一个位置，从而提供尽可能高的访问速度。此外，为了进一步提高访问速度并支持每秒数十万个请求，数据存储在新的存储桶类型中：Amazon S3 目录存储桶。有关更多信息，请参阅什么是 S3 Express One Zone？和目录桶。

您可以在 S3 Intelligent-Tiering 中存储具有不断变化或未知访问模式的数据，该分层可在访问模式发生变化时自动在四个访问层之间移动数据，从而优化存储成本。这四个访问层包括两个低延迟访问层（针对频繁和不频繁访问进行了优化），以及两个为异步访问很少访问的数据而设计的 opt-in archive 访问层。

有关更多信息，请参阅 使用 Amazon S3 存储类。

存储管理

Amazon S3 具有存储管理功能，您可以使用这些功能来管理成本、满足法规要求、减少延迟并保存数据的多个不同副本以满足合规性要求。

- S3 生命周期 - 配置生命周期配置以管理您的对象，并在整个生命周期内经济高效地存储这些对象。您可以将对象转换为其他 S3 存储类，也可以使其生命周期结束的对象过期。
- S3 对象锁定 - 可以在固定的时间段内或无限期地阻止删除或覆盖 Amazon S3 对象。可以使用对象锁定来帮助您满足需要一次写入多次读取（WORM）存储的法规要求，或只是添加另一个保护层来防止对象被更改和删除。
- S3 复制 - 将对象及其各自的元数据和对象标签复制到同一或不同的 AWS 区域 目标存储桶中的一个或多个目标存储桶，以减少延迟、合规性、安全性和其他使用案例。
- S3 批量操作 - 通过单个 S3 API 请求或在 Amazon S3 控制台中单击几次，大规模管理数十亿个对象。您可以使用批量操作来执行诸如复制、调用 AWS Lambda 函数和恢复数百万或数十亿对象这样的操作。

访问管理和安全性

Amazon S3 提供了用于审核和管理对存储桶和对象的访问的功能。默认情况下，S3 存储桶和对象都是私有的。您只能访问您创建的 S3 资源。要授予支持您特定使用案例的细粒度资源权限或审核 Amazon S3 资源的权限，您可以使用以下功能。

- S3 阻止公有访问 - 阻止对 S3 存储桶和对象的公有访问。默认情况下，“屏蔽公共访问权限”设置在存储桶级别处于开启状态。我们建议您将所有“屏蔽公共访问权限”设置保持为启用状态，除非您知道您

需要为您的特定使用案例关闭其中一个或多个设置。有关更多信息，请参阅 为 S3 存储桶配置屏蔽公共访问权限设置。

- AWS Identity and Access Management（IAM）– IAM 是一种 Web 服务，可帮助您安全地控制对 AWS 资源（包括 Amazon S3 资源）的访问。借助 IAM，您可以集中管理控制用户可访问哪些 AWS 资源的权限。可以使用 IAM 来控制谁通过了身份验证（准许登录）并获得授权（具有相应权限）来使用资源。
- 存储桶策略 - 使用基于 IAM 的策略语言为 S3 存储桶及其中的对象配置基于资源的权限。
- Amazon S3 接入点 – 使用专用访问策略配置命名网络端点，以便大规模管理对 Amazon S3 中共享数据集的访问。
- 访问控制列表（ACL）- 向授权用户授予单个存储桶和对象的读写权限。作为一般规则，我们建议您使用基于 S3 资源的策略（存储桶策略和接入点策略）或 IAM 用户策略进行访问控制，而不是 ACL。策略是一种简化、更灵活的访问控制选项。借助存储桶策略和接入点策略，您可以定义广泛适用于针对 Amazon S3 资源的所有请求的规则。有关何时使用 ACL 而不是基于资源的策略或 IAM 用户策略的特定情况的更多信息，请参阅使用 ACL 管理访问。
- S3 对象所有权 – 获取存储桶中每个对象的所有权，从而简化对存储在 Amazon S3 中的数据的访问管理。S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来禁用或启用 ACL。默认情况下，ACL 处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对数据的访问权限。
- 适用于 S3 的 IAM Access Analyzer - 评估和监控您的 S3 存储桶访问策略，确保这些策略仅提供对 S3 资源的预期访问权限。

数据处理

要转换数据并触发工作流以大规模自动执行各种其他处理活动，您可以使用以下功能。

- S3 Object Lambda - 您可以将自己的代码添加到 S3 GET、HEAD 和 LIST 请求中，以便在数据返回到应用程序时修改和处理数据。筛选行、动态调整图像大小、编辑机密数据等。
- 事件通知 - 当您的 S3 资源进行更改时，触发使用 Amazon Simple Notification Service (Amazon SNS)、Amazon Simple Queue Service (Amazon SQS) 和的工作流程和 AWS Lambda。

存储日志记录和监控

Amazon S3 提供日志记录和监控工具，您可以使用这些工具来监控和控制 Amazon S3 资源的使用情况。更多信息，请参阅监控工具。

自动监控工具

- Amazon S3 的 Amazon CloudWatch 指标- 跟踪 S3 资源的运行状况，并在估计费用达到用户定义的阈值时配置计费警报。
- AWS CloudTrail - 在 Amazon S3 中记录用户采取的行动、角色或 AWS 服务。CloudTrail 日志为您提供了 S3 存储桶级别和对象级操作的详细 API 跟踪。

手动监控工具

- 服务器访问日志- 详细地记录对存储桶提出的各种请求。您可以使用服务器访问日志对许多使用案例进行安全和访问审计，了解客户群或了解您的 Amazon S3 账单。
- AWS Trusted Advisor - 通过使用 AWS 最佳实践检查以确定优化 AWS 基础架构、提高安全性和性能、降低成本以及监控服务配额。然后，您可以按照建议优化服务和资源。

分析和见解

Amazon S3 提供的功能可帮助您了解存储使用情况，从而使您能够更好地了解、分析和大规模优化存储。

- Amazon S3 Storage Lens - 了解、分析和优化您的存储。S3 Storage Lens 存储统计管理工具提供了超过 60 个使用率和活动指标以及交互式控制面板，用于汇总整个组织、特定客户、AWS 区域、存储桶或前缀的数据。
- 存储类分析- 分析存储访问模式，以决定何时需要将数据移动到更经济高效的存储类。
- 带清单报告的 S3 清单- 审核和报告对象及其相应的元数据，并配置其他 Amazon S3 功能，在清单报告中采取措施。例如，您可以报告对象的复制和加密状态。有关清单报告中每个对象可用的所有元数据的列表，请参阅 Amazon S3 清单。

强一致性

Amazon S3 为所有 AWS 区域 中的 Amazon S3 存储桶中对象的 PUT 和 DELETE 请求提供了强大的先写后读一致性。这个行为既适用于到新对象的写入，也适用于覆盖现有对象的 PUT 和 DELETE。此外，针对 Amazon S3 Select、Amazon S3 访问控制列表 (ACL)、Amazon S3 对象标签和对象元数据（例如 HEAD 对象）的读取操作具有严格一致性。有关更多信息，请参阅 Amazon S3 数据一致性模型。

Amazon S3 的工作原理

Amazon S3 是一种对象存储服务，可将数据以对象形式存储在存储桶中。对象指的是一个文件和描述该文件的任何元数据。存储桶是对象的容器。

要将数据存储在 Amazon S3 中，您需要先创建存储桶，然后指定存储桶名称和 AWS 区域。然后，您将数据作为 Amazon S3 中的对象上传到该存储桶。每个对象都带有密钥（或键名称），它是存储桶中对象的唯一标识符。

S3 提供了一些功能，您可以配置这些功能以支持您的特定使用案例。例如，您可以使用 S3 版本控制将对象的多个版本保持在同一个存储桶中，这允许您恢复意外删除或覆盖的对象。

存储桶及其中的对象是私有的，只有在您明确授予访问权限时才可以访问。您可以使用存储桶策略、AWS Identity and Access Management（IAM）策略、访问控制列表（ACL）和 S3 接入点管理访问。

主题

- 存储桶
- 对象
- 键
- S3 版本控制
- 版本 ID
- 存储桶策略
- S3 接入点
- 访问控制列表（ACL）
- 区域

存储桶

存储桶是 Amazon S3 中用于存储对象的容器。您可以在存储桶中存储任意数量的对象，并且账户中最多可以有 100 个存储桶。要请求提高限额，请访问服务限额控制台。

每个对象都储存在一个存储桶中。例如，如果名为 `photos/puppy.jpg` 的对象存储在美国西部（俄勒冈州）区域的 `amzn-s3-demo-bucket` 存储桶中，则可使用 URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg` 对该对象进行寻址。有关更多信息，请参阅访问存储桶。

创建存储桶时，您可以输入存储桶名称，然后选择AWS 区域存储桶将驻留的位置。创建存储桶后，无法更改存储桶或区域的名称。存储桶名称必须遵循存储桶命名规则。您也可以将存储桶配置为使用 S3 版本控制或其他存储管理功能。

存储桶还：

- 在最高级别组织 Amazon S3 命名空间。
- 标识负责存储和数据传输费用的账户。
- 提供访问控制选项，例如存储桶策略、访问控制列表（ACL）和 S3 接入点，可用于管理对 Amazon S3 资源的访问。
- 用作使用情况报告的聚合单元。

有关存储桶的更多信息，请参阅 存储桶概述。

对象

对象是 Amazon S3 中存储的基础实体。对象由对象数据和元数据组成。元数据是一组描述对象的名称-值对。这些对值包括一些默认元数据（如上次修改日期）和标准 HTTP 元数据（如 `Content-Type`）。您还可以在存储对象时指定自定义元数据。

存储桶中的对象由密钥（名称）和版本 ID唯一标识（如果在存储桶上启用了 S3 版本控制）。有关对象的更多信息，请参阅 Amazon S3 对象概述。

键

对象密钥（或密钥名称）是指存储桶中对象的唯一标识符。存储桶内的每个对象都只能有一个键。存储桶、对象密钥和可选版本 ID 的组合（如果为存储桶启用了 S3 版本控制）唯一标识每个对象。因此，您可以将 Amazon S3 看作“存储桶 + 键 + 版本”与对象本身之间的基本数据映射。

将 Web 服务端点、存储桶名称、密钥和版本（可选）组合在一起，可唯一地寻址 Amazon S3 中的每个对象。例如，在 URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg` 中，*`amzn-s3-demo-bucket`* 是存储桶的名称，`photos/puppy.jpg` 是密钥。

有关对象键的更多信息，请参阅 创建对象键名称。

S3 版本控制

您可以使用 S3 版本控制功能将对象的多个变量保留在同一存储桶中。使用 S3 版本控制功能，您可以保留、检索和恢复存储桶中的各个版本。您能够轻松从用户意外操作和应用程序故障中恢复数据。

有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

版本 ID

在存储桶中启用 S3 版本控制时，Amazon S3 会为添加到存储桶中的每个对象生成唯一的版本ID。启用版本控制时存在于存储桶中的对象的版本 ID 为null。如果使用其他操作修改这些（或任何其他）对象，例如 CopyObject 和 PutObject 时，新对象将获得唯一的版本 ID。

有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

存储桶策略

存储桶策略是基于资源的 AWS Identity and Access Management (IAM) 策略，您可以使用该策略向存储桶及其中对象授予访问权限。只有存储桶拥有者才能将策略与存储桶关联。附加到存储桶的权限适用于存储桶拥有者拥有的存储桶中所有对象。存储桶策略的大小限制为 20 KB。

存储桶策略使用基于 JSON 的访问策略语言，该语言是跨平台的标准语言 AWS。您可以使用存储桶策略添加或拒绝存储桶中对象的权限。存储桶策略根据策略中的元素允许或拒绝请求，包括请求者、S3 操作、资源以及请求的方面或条件（例如，用于发出请求的 IP 地址）。例如，您可以创建一个存储桶策略，该策略授予跨账户将对象上传到 S3 存储桶的权限，同时确保存储桶拥有者对上传的对象拥有完全控制权。有关更多信息，请参阅 Amazon S3 存储桶策略的示例。

在存储桶策略中，您可以在 Amazon 资源名称（ARN）和其他值上使用通配符来授予对对象子集的权限。例如，您可以控制对以通用前缀或以给定扩展名结尾的对象组的访问，例如 .html。

S3 接入点

Amazon S3 接入点被命名为网络端点，其专用访问策略描述了如何使用该端点访问数据。接入点附加到存储桶，您可以使用这些存储桶执行 S3 对象操作（如 GetObject 和 PutObject）。接入点可简化对 Amazon S3 中的共享数据集的大规模数据访问管理。

每个接入点都有自己的接入点策略。您还可以为每个接入点配置阻止公有访问设置。为了限制 Amazon S3 数据访问提供网络，您可以将任何接入点配置为仅接受来自私有云（VPC）的请求。

有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。

访问控制列表（ACL）

您可以使用 ACL 向已授权的用户授予对单个存储桶和对象的读写权限。每个存储桶和对象都有一个作为子资源而附加的 ACL。ACL 定义了哪些 AWS 账户 或组将被授予访问权限以及访问的类型。ACL 是一种访问控制机制，其早于 IAM。有关 ACL 的更多信息，请参阅 访问控制列表 (ACL) 概述。

S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权和禁用或启用 ACL。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对这些对象的访问权限。

Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有需要单独控制每个对象的访问权限的特殊情况。禁用 ACL 后，您可以使用策略来控制对存储桶中所有对象的访问权限，无论是谁将对象上传到您的存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

区域

您可以选择一个 AWS 区域 供 Amazon S3 存储您创建的存储桶。您可以选择一个区域，以便优化延迟、尽可能降低成本或满足法规要求。存储在 AWS 区域 中的对象永远不会离开该区域，除非显式地将它们传输或复制到另一个区域。例如，在欧洲（爱尔兰）区域存储的对象将一直留在欧洲。

> **Note**
>
> 您只能在已为账户启用的 AWS 区域 中访问 Amazon S3 及其功能。有关启用区域以创建和管理 AWS 资源的更多信息，请参阅《AWS 一般参考》中的管理 AWS 区域。

有关 Amazon S3 区域和端点的列表，请参阅《AWS 一般参考》中的区域和端点。

Amazon S3 数据一致性模型

Amazon S3 为所有 AWS 区域 中的 Amazon S3 存储桶中对象的 PUT 和 DELETE 请求提供了强大的先写后读一致性。此行为既适用于对新对象的写入，也适用于覆盖现有对象的 PUT 请求和 DELETE 请求。此外，Amazon S3 Select、Amazon S3访问控制列表 (ACL)、Amazon S3对象标记和对象元数据（例如，HEAD对象）上的读取操作非常一致。

单个键的更新是原子更新。例如，如果您从一个线程对现有密钥执行 PUT 请求，并同时从另一个线程对同一密钥执行 GET 请求，则您将获得旧数据或新数据，但绝不会获得部分或损坏的数据。

Amazon S3 通过在 AWS 数据中心内的多个服务器之间复制数据，从而实现高可用性。如果 PUT 请求成功，则数据已安全存储。在收到成功的 PUT 响应后启动的任何读取（GET 或 LIST 请求）都将返回 PUT 请求写入的数据。以下是此行为的示例：

- 这是一个过程，它将一个新对象写入 Amazon S3，并立即列出其存储桶内的密钥。新对象显示在列表中。

- 这是一个过程，会替换一个现有的对象，并立即尝试读取此对象。Amazon S3 返回新数据。
- 这是一个过程，会删除一个现有的对象，并立即尝试读取此对象。Amazon S3 不会返回任何数据，因为对象已被删除。
- 这是一个过程，会删除一个现有的对象，并立即列出其存储桶内的键。该对象不会显示在列表中。

> **Note**
>
> - Amazon S3 不支持对并发写入器进行对象锁定。如果同时对同一键发出两个 PUT 请求，则以带有最新时间戳的请求为准。如果这会导致问题，您需要在应用程序中创建对象锁定机制。
> - 更新是基于键的。无法跨键值实现原子更新。例如，无法根据一个键值的更新对另一个键值进行更新，除非将此功能设计到应用程序中。

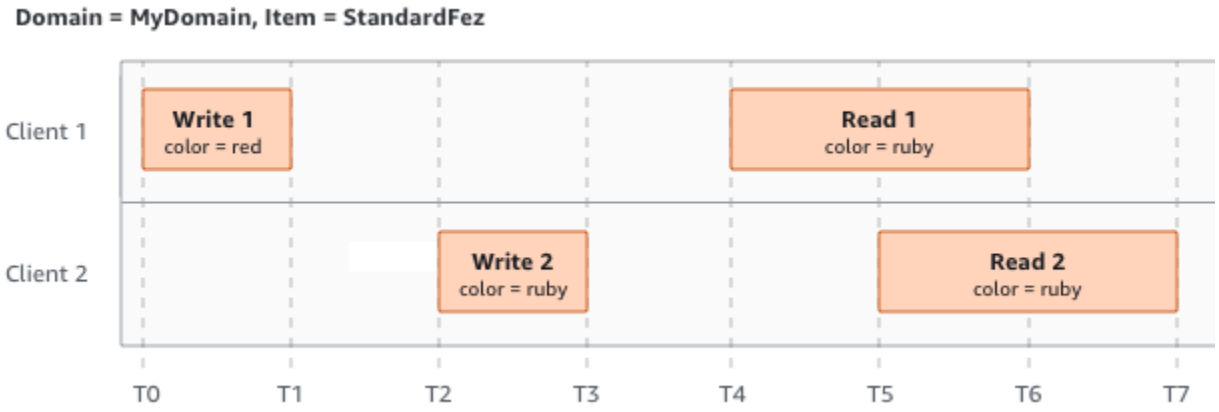
存储桶配置具有最终一致性模型。具体而言，这意味着：

- 如果您删除存储桶，然后立即列出所有存储桶，则所删除的存储桶可能仍会显示在列表中。
- 如果您首次对存储桶启用版本控制，则可能需要很短的时间即可完全传播更改。我们建议您在启用版本控制后等待 15 分钟，然后再对存储桶中的对象发出写入操作（PUT 或 DELETE 请求）。

并发应用程序

本部分提供了当多个客户端向同一项目写入时，Amazon S3 应采取的行为示例。

在本示例中，W1 (写入 1) 和 W2 (写入 2) 会在 R1 (读取 1) 和 R2 (读取 2) 启动之前完成。由于 S3 具有严格一致性，因此 R1 和 R2 都会返回 `color = ruby`。



在下一个示例中，W2 不会在 R1 启动之前完成。因此，R1 可能会返回 `color = ruby` 或 `color = garnet`。但是，由于 W1 和 W2 在 R2 开始之前完成，因此 R2 会返回 `color = garnet`。

在上一个示例中，W2 在 W1 收到确认之前开始。因此，这些写入被视为并发写入。Amazon S3 在内部使用 last-writer-wins 语义来确定哪个写入优先。但是，由于网络延迟等各种因素，无法预测 Amazon S3 接收请求的顺序和应用程序接收确认的顺序。例如，W2 可能由同一区域中的 Amazon EC2 实例启动，而 W1 可能由距离更远的主机启动。确定最终值的最佳方法是在确认两次写入操作后执行读取。

相关服务

将数据加载到 Amazon S3 中之后，您可以将数据用于其他 AWS 服务。以下是您可能最常用使用的服务：

- Amazon Elastic Compute Cloud (Amazon EC2) – 在 AWS Cloud中提供安全可扩展的计算容量。使用 Amazon EC2 可避免前期的硬件投入，因此您能够更快地开发和部署应用程序。您可以使用 Amazon EC2 启动所需数量的虚拟服务器，配置安全性和联网以及管理存储。

- Amazon EMR – 帮助企业、研究人员、数据分析师和开发人员轻松、经济实惠地处理海量数据。Amazon EMR 使用 Amazon EC2 和 Amazon S3 的 Web 规模基础设施上运行的托管 Hadoop 框架。
- AWSSnow 系列 - 帮助那些需要在严格、非数据中心环境中以及在缺乏一致网络连接的位置运行操作的客户。您可以使用 AWS Snow Family 设备可在本地经济高效地访问 AWS Cloud 在互联网连接可能不是一个选项的地方。
- AWS Transfer Family - 为使用安全外壳 (SSH) 文件传输协议 (SFTP)、通过 SSL (FTPS) 的文件传输协议 (FTPS) 和文件传输协议 (FTP) 直接进出 Amazon S3 或 Amazon Elastic File System (Amazon EFS) 的文件传输提供完全托管支持。

访问 Amazon S3

您可以通过以下任何方式使用 Amazon S3：

AWS Management Console

控制台是基于 Web 的用户界面，用于管理 Amazon S3 和 AWS 资源。如果您已注册 AWS 账户 账户，您可以通过登录 AWS Management Console 并从 AWS Management Console 主页中选择 S3 来访问 Amazon S3 控制台。

AWS Command Line Interface

可以使用 AWS 命令行工具，在系统的命令行中发出命令来执行 AWS （包括 S3）任务。

AWS Command Line Interface (AWS CLI) 针对大量 AWS 服务 提供了相关命令。AWS CLI 在 Windows、macOS 和 Linux 上受支持。要开始使用，请参阅 AWS Command Line Interface 用户指南。有关 Amazon S3 命令的更多信息，请参阅 AWS CLI 一般参考中的 s3api 和 s3control。

AWS SDK

AWS 提供的 SDK（软件开发工具包）包含各种编程语言和平台（Java、Python、Ruby、.NET、iOS、Android 等）的库和示例代码。AWS SDK 提供便捷的方式来创建对 S3 和 AWS 的编程访问。Amazon S3 是一项 REST 服务。您可以使用 AWS SDK 库向 Amazon S3 发送请求，该库包装了底层 Amazon S3 REST API 并简化了编程任务。例如，SDK 负责计算签名、加密签名请求、管理错误和自动重试请求等任务。有关 AWS SDK 的信息（包括如何下载及安装），请参阅 适用于 AWS 的工具 。

与 Amazon S3 的每一次交互都是经身份验证的或匿名的。如果您使用 AWS SDK，库根据您提供的密钥计算用于身份验证的签名。有关如何向 Amazon S3 发出请求的更多信息，请参阅 提出请求。

Amazon S3 REST API

Amazon S3 架构的设计与编程语言无关，使用 AWS 支持的接口来存储和检索对象。您可以访问 S3 和 AWS 以编程方式使用 Amazon S3 REST API。REST API 是面向 Amazon S3 的 HTTP 接口。借助 REST API，您可以使用标准的 HTTP 请求创建、提取和删除存储桶和对象。

要使用 REST API，您可以借助任何支持 HTTP 的工具包。只要对象是匿名可读的，您甚至可以使用浏览器来提取它们。

REST API 使用标准的 HTTP 标头和状态代码，以使标准的浏览器和工具包按预期工作。在某些区域中，我们向 HTTP 添加了功能 (例如，我们添加了标头来支持访问控制)。在这些情况下，我们已尽最大努力使添加的新功能与标准的 HTTP 使用样式相匹配。

如果您在应用程序中直接调用 REST API，您必须编写代码来计算签名并将它添加到请求中。有关如何向 Amazon S3 发出请求的更多信息，请参阅 提出请求。

> **Note**
>
> HTTP 上的 SOAP API 支持已弃用，但是仍可在 HTTPS 上使用。SOAP 不支持较新的 Amazon S3 特征。我们建议您使用 REST API 或 AWS SDK。

为 Amazon S3 支付费用

Amazon S3 定价的设计可让您不必为应用程序的存储需求制定计划。大多数存储提供商都要求您购买预定量的存储和网络传输容量。在这种情况下，如果超过此容量，就会关闭您的服务或对您收取高额的超容量费用。如果没有超过此容量，又要按全部容量支付使用费用。

Amazon S3 仅按照您的实际使用容量收费，没有任何隐性收费和超容量收费。此模型为您提供了一种可变成本服务，它可以随着您的业务增长而增长，同时为您提供 AWS 基础架构的成本优势。有关更多信息，请参阅 Amazon S3 定价。

在注册 AWS 时，将在 AWS 中为您的 AWS 账户 自动注册所有服务，包括 Amazon S3。不过，您只需为使用的服务付费。如果您是 Amazon S3 的新客户，则可以免费开始使用 Amazon S3。有关更多信息，请参阅 AWS免费套餐。

若要查看您的账单，请转到 AWS Billing and Cost Management 控制台中的账单和成本管理控制面板。要了解关于AWS 账户 账单的更多信息，请参阅 AWS Billing 用户指南。如果您有关于 AWS 账单和 AWS 账户的问题，请联系 AWS 支持。

PCI DSS 合规性

Amazon S3 支持由商家或服务提供商处理、存储和传输信用卡数据，而且已经验证符合支付卡行业 (PCI) 数据安全标准 (DSS)。有关 PCI DSS 的更多信息，包括如何请求 AWS PCI Compliance Package 的副本，请参阅 PCI DSS 第 1 级。

Amazon S3 入门

您可以通过使用存储桶和对象来开始使用 Amazon S3。存储桶是对象的容器。对象指的是一个文件和描述该文件的任何元数据。

要在 Amazon S3 中存储对象，您需要创建存储桶，然后将该对象上传到存储桶。当对象位于存储桶时，您可以将其打开、下载并移动它。当您不再需要对象或存储桶时，可以清理您的资源。

使用 Amazon S3，您只需按实际用量付费。有关 Amazon S3 特征和定价的更多信息，请参阅 Amazon S3。如果您是 Amazon S3 的新客户，则可以免费开始使用 Amazon S3。有关更多信息，请参阅 AWS 免费套餐。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅什么是 S3 Express One Zone？和目录桶。

视频：Amazon S3 入门

先决条件

开始之前，请确认您已完成先决条件：设置 Amazon S3中的步骤。

主题

先决条件：设置 Amazon S3

在注册 AWS 时，将在 AWS 中为您的 AWS 账户 自动注册所有服务，包括 Amazon S3。您只需为使用的服务付费。

使用 Amazon S3，您只需按实际用量付费。有关 Amazon S3 特征和定价的更多信息，请参阅 Amazon S3。如果您是 Amazon S3 的新客户，则可以免费开始使用 Amazon S3。有关更多信息，请参阅 AWS 免费套餐。

要设置 Amazon S3，请使用以下部分中的步骤。

注册 AWS 并设置 Amazon S3 时，您可以选择在 AWS Management Console 中更改显示语言。有关更多信息，请参阅《AWS Management Console 入门指南》中的更改 AWS Management Console 的语言。

主题

注册 AWS 账户

如果您还没有 AWS 账户，请完成以下步骤来创建一个。

注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册 AWS 账户时，系统将会创建一个 AWS 账户根用户。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

注册过程完成后，AWS 会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册 AWS 账户 后，请保护好您的 AWS 账户根用户，启用 AWS IAM Identity Center，并创建一个管理用户，以避免使用根用户执行日常任务。

保护您的 AWS 账户根用户

1. 选择根用户并输入您的 AWS 账户电子邮件地址，以账户拥有者身份登录 AWS Management Console。在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅《IAM 用户指南》中的为 AWS 账户根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关如何使用 IAM Identity Center 目录 作为身份源的教程，请参阅《AWS IAM Identity Center 用户指南》中的使用默认的 IAM Identity Center 目录 配置用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  要获取使用 IAM Identity Center 用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

步骤 1：创建您的第一个 S3 存储桶

注册 AWS 后，您即可使用 AWS Management Console 在 Amazon S3 中创建存储桶了。Amazon S3 中的每个对象都存储在存储桶中。必须先创建一个存储桶，然后才能在 Amazon S3 中存储数据。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅什么是 S3 Express One Zone？和目录桶。

> **Note**
>
> 您无需为创建存储桶付费。只有将对象存储到存储桶中以及从存储桶传入和传出对象才需要付费。仿照本指南中的示例操作产生的费用非常少（不到 1 USD）。有关存储费用的更多信息，请参阅 Amazon S3 定价。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要在其中创建存储桶的区域。

   > **Note**
   >
   > 要最大程度地减少延迟和成本以及满足法规要求，请选择一个靠近您的区域。在某一区域存储的对象将一直留在该区域，除非您特意将其转移到其他区域。有关 Amazon S3 AWS 区域的列表，请参阅《Amazon Web Services 一般参考》中的 AWS 服务 端点。

3. 在左侧导航窗格中，选择存储桶。
4. 选择创建存储桶。

   此时将打开创建存储桶页面。
5. 在常规配置下，查看将在其中创建存储桶的 AWS 区域。
6. 在存储桶类型下，选择通用。
7. 对于存储桶名称，请输入存储桶的名称。

存储桶名称必须满足以下要求：

- 在分区中是唯一的。分区是一组区域。AWS 目前有三个分区：aws（标准区域）、aws-cn（中国区域）和 aws-us-gov（AWS GovCloud (US) Regions）。
- 长度必须介于 3 到 63 个字符之间。
- 只能由小写字母、数字、句点（.）和连字符（-）组成。为了获得最佳兼容性，我们建议您避免在存储桶名称中使用句点（.），但仅用于静态网站托管的存储桶除外。
- 以字母或数字开头和结尾。

创建存储桶后，便无法再更改其名称。有关给存储桶命名的更多信息，请参阅存储桶命名规则。

> **Important**
>
> 避免在存储桶名称中包含敏感信息，如账号。存储桶名称会显示在指向存储桶中的对象的 URL 中。

8. AWS Management Console可让您将现有存储桶的设置复制到新存储桶。如果您不想复制现有存储桶的设置，请跳到下一步。

> **Note**
>
> 此选项：
>
> - 在 AWS CLI 中不可用，仅在控制台中可用
> - 不可用于目录存储桶
> - 不会将存储桶策略从现有存储桶复制到新存储桶

要复制现有存储桶的设置，请在从现有存储桶复制设置下，选择选择存储桶。将打开选择存储桶窗口。找到包含您要复制的设置的存储桶，然后选择选择存储桶。选择存储桶窗口关闭，创建存储桶窗口重新打开。

在从现有存储桶复制设置下，您现在将看到所选存储桶的名称。您还将看到还原默认值选项，您可以使用该选项移除复制的存储桶设置。在创建存储桶页面上查看其余存储桶设置。您将看到它们现在与您选择的存储桶的设置相匹配。您可以跳到最后一步。

9. 在 Object Ownership（对象所有权）下方，要禁用或启用 ACL，并控制上传到存储桶中的对象的所有权，请选择以下设置之一：

   已禁用 ACL

   - 强制存储桶拥有者（默认）– ACL 已禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响对 S3 存储桶中数据的访问权限。存储桶专门使用策略来定义访问控制。

     默认情况下，ACL 处于禁用状态。Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有必须单独控制每个对象的访问权限的特殊情况。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

   已启用 ACL

   - Bucket owner preferred（首选存储桶拥有者）— 存储桶拥有者拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 写入存储桶的新对象。

     如果应用首选存储桶拥有者设置，以要求所有 Amazon S3 上传都包含 `bucket-owner-full-control` 标准 ACL，则可以添加存储桶策略，该策略只允许使用此 ACL 上传对象。
   - 对象编写者— 该 AWS 账户上传对象拥有该对象，对其拥有完全控制权，并且可以通过 ACL 授予其他用户访问该对象的权限。

   > **Note**
   >
   > 默认设置为强制存储桶拥有者。要应用默认设置并将 ACL 保持为禁用状态，只需要 `s3:CreateBucket` 权限。要启用 ACL，您必须具有 `s3:PutBucketOwnershipControls` 权限。

10. 在此存储桶的屏蔽公共访问权限设置中，请选择要应用于存储桶的屏蔽公共访问权限设置。

    默认情况下，启用所有四个“屏蔽公共访问权限”设置。我们建议您将所有设置保持为启用状态，除非您知道您需要为您的特定使用案例关闭其中一个或多个设置。有关屏蔽公共访问权限的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

> **Note**
>
> 要启用所有“屏蔽公共访问权限”设置，只需要 `s3:CreateBucket` 权限。要关闭任何“屏蔽公共访问权限”设置，您必须拥有 `s3:PutBucketPublicAccessBlock` 权限。

11. （可选）在 Bucket Versioning（存储桶版本控制）下，您可以选择是否要在存储桶中保留对象的变体。有关版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。

    要在存储桶上禁用或启用版本控制，请选择 Disable（禁用）或 Enable（启用）。

12. （可选）在 Tags（标签）下，您可以选择向存储桶添加标签。标签是用于对存储进行分类的键/值对。

    要添加存储桶标签，请输入 Key（键），并（可选）输入 Value（值），然后选择 Add Tag（添加标签）。

13. 在默认加密下，请选择编辑。

14. 要配置默认加密，请在加密类型下，选择以下选项之一：

    - Amazon S3 托管密钥（SSE-S3）
    - AWS Key Management Service 密钥（SSE-KMS）

    > **Important**
    >
    > 如果您将 SSE-KMS 选项用于默认加密配置，则您将受到 AWS KMS 的每秒请求数（RPS）限额限制。有关 AWS KMS 限额以及如何请求增加限额的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的限额。

    存储桶和新对象使用具有 Amazon S3 托管密钥的服务器端加密进行加密，作为加密配置的基本级别。有关默认加密的更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

    有关使用 Amazon S3 服务器端加密对数据进行加密的更多信息，请参阅使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

15. 如果选择了 AWS Key Management Service 密钥（SSE-KMS），则执行以下操作：

    a. 在 AWS KMS 密钥下，通过以下方式之一指定您的 KMS 密钥：

- 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 密钥中进行选择，并从可用密钥的列表中选择您的 KMS 密钥。

  AWS 托管式密钥 (aws/s3) 和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。
- 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
- 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

  有关创建 AWS KMS key 的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

> **Important**
>
> 您只能使用与存储桶所在相同的 AWS 区域中可用的 KMS 密钥。Amazon S3 控制台仅列出与存储桶位于同一区域中的前 100 个 KMS 密钥。要使用未列出的 KMS 密钥，您必须输入 KMS 密钥 ARN。如果您希望使用其他账户拥有的 KMS 密钥，则必须首先有权使用该密钥，然后必须输入相应的 KMS 密钥 ARN。有关 KMS 密钥的跨账户权限的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建其他账户可以使用的 KMS 密钥。有关 SSE-KMS 的更多信息，请参阅使用 AWS KMS (SSE-KMS) 指定服务器端加密。
>
> 在 Amazon S3 中使用 AWS KMS key 进行服务器端加密时，您必须选择对称加密 KMS 密钥。Amazon S3 仅支持对称加密 KMS 密钥，而不支持非对称 KMS 密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的确定对称和非对称 KMS 密钥。

有关创建 AWS KMS key 的更多信息，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。有关将 AWS KMS 与 Amazon S3 结合使用的更多信息，请参阅使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。

b. 将存储桶配置为使用 SSE-KMS 进行默认加密时，您还可以启用 S3 存储桶密钥。S3 存储桶密钥可通过减少从 Amazon S3 到 AWS KMS 的请求流量，降低加密成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

   要使用 S3 存储桶密钥，请在 Bucket Key（存储桶密钥）下，选择 Enable（启用）。

16. （可选）如果要启用 S3 对象锁定，请执行以下操作：

    a. 选择高级设置。

    > **Important**
    >
    > 启用对象锁定还会启用存储桶的版本控制。启用后，必须配置对象锁定默认保留和法律保留设置，以保护新对象不被删除或覆盖。

    b. 如果要启用对象锁定，请选择 Enable（启用），阅读出现的警告，并予以确认。

    有关更多信息，请参阅 使用 S3 对象锁定。

    > **Note**
    >
    > 要创建启用对象锁定的存储桶，您必须具有以下权限：`s3:CreateBucket`、`s3:PutBucketVersioning` 和 `s3:PutBucketObjectLockConfiguration`。

17. 选择创建存储桶。

您已在 Amazon S3 中完成存储桶创建。

下一步

要将对象添加到您的存储桶，请参阅步骤 2：将对象上传到存储桶。

步骤 2：将对象上传到存储桶

在 Amazon S3 中创建存储桶后，您就可以将对象上传到存储桶了。对象可以是任何类型的文件：文本文件、图片、视频等等。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和 目录桶。

将对象上传到存储桶

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets (存储桶) 列表中，选择要将对象上传到的存储桶的名称。
3. 在存储桶的对象选项卡上，选择上传。
4. 在 Files and folders (文件和文件夹) 下，选择 Add files (添加文件)。
5. 选择要上传的文件，然后选择打开。
6. 选择上传。

您已成功将对象上传到存储桶。

下一步

要查看您的对象，请参阅 步骤 3：下载对象。

步骤 3：下载对象

当您将对象上传到存储桶后，您可以查看有关对象的信息并将对象下载到本地计算机上。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和 目录桶。

使用 S3 控制台

本部分介绍如何使用 Amazon S3 控制台从 S3 存储桶下载对象。

> **Note**
>
> - 一次只能下载一个对象。
> - 如果您使用 Amazon S3 控制台下载对象的键名以句点 (.) 结尾，则该句点将从下载对象的键名中删除。要保留所下载对象名称末尾的句点，您必须使用 AWS Command Line Interface (AWS CLI)、AWS SDK 或 Amazon S3 REST API。

从 S3 存储桶下载对象

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要从中下载对象的存储桶的名称。
3. 您可以使用以下任一方式从 S3 存储桶下载对象：
   - 选中对象旁边的复选框，然后选择下载。如果您要将对象下载到特定文件夹，请在操作菜单中，选择下载为。
   - 如果要下载对象的特定版本，请打开显示版本（位于搜索框旁边）。选中所需对象版本旁边的复选框，然后选择下载。如果您要将对象下载到特定文件夹，请在操作菜单中，选择下载为。

您已成功下载您的对象。

下一步

要在 Amazon S3 中复制并粘贴对象，请参阅步骤 4：将对象复制到文件夹。

步骤 4：将对象复制到文件夹

您已向存储桶添加一个对象并已下载该对象。现在，您可以创建一个文件夹，复制对象并将其粘贴到此文件夹中。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅什么是 S3 Express One Zone？和目录桶。

将对象复制到文件夹中

1. 在存储桶列表中，选择您的存储桶名称。
2. 选择创建文件夹并配置新文件夹：
   a. 输入文件夹名称（例如，`favorite-pics`）。
   b. 对于文件夹加密设置，选择禁用。

c. 选择保存。

3. 导航到包含待复制对象的 Amazon S3 存储桶或文件夹。
4. 选中要复制的对象名称左侧的复选框。
5. 选择操作，然后从显示的选项列表中选择复制。

   或者，从右上角的选项中选择复制。
6. 选择目标文件夹：

   a. 选择浏览 S3。

   b. 选择文件夹名称左侧的选项按钮。

      要导航到文件夹并选择子文件夹作为目标，请选择文件夹名称。

   c. 选择选择目标。

   目标文件夹的路径显示在目标框中。在目标中，您可以依次输入目标路径，例如 s3://*bucket-name*/*folder-name*/。
7. 在右下角，选择复制。

   Amazon S3 会将对象复制到目标文件夹。

下一步

要删除 Amazon S3 中的对象和存储桶，请参阅步骤 5：删除对象和存储桶。

步骤 5：删除对象和存储桶

当您不再需要对象或存储桶时，我们建议您删除它们以防止发生进一步的费用。如果您将此入门演练作为练习完成，并且不打算使用您的存储桶或对象，我们建议您删除存储桶和对象，从而不再产生费用。

在删除存储桶之前，请清空存储桶或删除存储桶中的对象。删除对象和存储桶后，它们将不再可用。

如果要继续使用相同的存储桶名称，我们建议您删除对象或清空存储桶，但不要删除存储桶。删除存储桶后，该名称可供重复使用。但是，在您有机会重复使用此名称之前，其他 AWS 账户可能会创建具有相同名称的存储桶。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅什么是 S3 Express One Zone？和目录桶。

主题

删除对象

如果您希望选择要删除的对象而不清空存储桶中的所有对象，则可以删除单个对象。

1. 在存储桶列表中，选择要从中删除对象的存储桶的名称。
2. 选择您要删除的对象。
3. 从右上角的选项中选择删除。
4. 在删除对象页面上，键入 **`delete`** 以确认删除您的对象。
5. 选择删除对象。

清空存储桶

如果您计划删除存储桶，则必须首先清空存储桶，这会删除存储桶中的所有对象。

清空存储桶

1. 在存储桶列表中，选择要清空的存储桶，然后选择清空。
2. 要确认您要清空存储桶并删除其中的所有对象，请在 Empty bucket (清空存储桶) 中，键入 **`permanently delete`**。

> **Important**
>
> 无法撤消清空存储桶操作。当正在执行清空存储桶操作时添加到存储桶的对象将被删除。

3. 要清空存储桶并删除其中的所有对象，并选择 Empty (清空)。

   Empty bucket: Status (清空存储桶：状态) 页面打开，您可以使用该页面来查看失败和成功的对象删除操作的摘要。

4. 要返回到存储桶列表，请选择退出。

删除存储桶

清空存储桶或从存储桶中删除所有对象后，您可以删除存储桶。

1. 要删除存储桶，请在存储桶列表中选择该存储桶。
2. 选择删除。
3. 要确认删除，请在 Delete bucket (删除存储桶) 中，键入存储桶的名称。

   > **Important**
   >
   > 无法撤消删除存储桶的操作。存储桶名称是唯一的。如果您删除存储桶，则其他 AWS 用户可以使用该名称。如果您希望继续使用相同的存储桶名称，请不要删除该存储桶。相反，清空并保留存储桶。

4. 要删除您的存储桶，请选择删除存储桶。

后续步骤

在前面的示例中，您了解了如何执行一些基本的 Amazon S3 任务。

下列主题介绍了各种可促使您深入了解 Amazon S3，进而在应用程序中实施服务的方式。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅什么是 S3 Express One Zone？和目录桶。

主题

- 了解常见使用案例
- 控制对存储桶和对象的访问

- 管理和监控您的存储
- 用 Amazon S3 进行开发
- 从教程中学习
- 探索培训和支持

了解常见使用案例

您可以使用 Amazon S3 支持您的具体使用案例。AWS 解决方案库和 AWS 博客提供了具体使用案例的信息和教程。以下是 Amazon S3 的一些常见使用案例：

- Backup and storage（备份和存储）- 使用 Amazon S3 存储管理功能来管理成本、满足法规要求、减少延迟并保存多个不同的数据拷贝以满足合规性要求。
- 应用程序托管 - 部署、安装和管理可靠、高度可扩展和低成本的 Web 应用程序。例如，可以配置您的 Amazon S3 存储桶来承载一个静态网站。有关更多信息，请参阅 使用 Amazon S3 托管静态网站。
- 媒体托管 - 构建托管视频、照片或音乐上载和下载的高可用性基础设施。
- 软件传输 - 托管可供客户下载的软件应用程序。

控制对存储桶和对象的访问

Amazon S3 提供了各种安全功能和工具。有关概述，请参阅访问控制。

默认情况下，S3 存储桶和对象都是私有的。您只能访问您创建的 S3 资源。您可以使用以下功能授予支持具体使用案例的细粒度资源权限，或审核您的 Amazon S3 资源的权限。

- S3 阻止公有访问 - 阻止对 S3 存储桶和对象的公有访问。默认情况下，“屏蔽公共访问权限”设置在存储桶级别处于开启状态。
- AWS Identity and Access Management（IAM）身份 – 使用 IAM 或 AWS IAM Identity Center 在 AWS 账户 中创建 IAM 身份，以管理对 Amazon S3 资源的访问。例如，您可以将 IAM 用于 Amazon S3，以控制用户或用户组对您的 AWS 账户拥有 Amazon S3 存储桶的访问类型。有关 IAM 身份和最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 身份（用户、用户组和角色）。
- 桶策略 - 使用基于 IAM 的策略语言为 S3 桶及其中的对象配置基于资源的权限。
- 访问控制列表（ACL）- 向授权用户授予单个存储桶和对象的读写权限。作为一般规则，我们建议您使用基于 S3 资源的策略（存储桶策略和接入点策略）或 IAM 用户策略进行访问控制，而不是

ACL。策略是一种简化、更灵活的访问控制选项。借助存储桶策略和接入点策略，您可以定义广泛适用于针对 Amazon S3 资源的所有请求的规则。有关何时使用 ACL 而不是基于资源的策略或 IAM 用户策略的特定情况的更多信息，请参阅Amazon S3 的身份和访问管理。

- S3 对象所有权 – 获取存储桶中每个对象的所有权，从而简化对存储在 Amazon S3 中的数据的访问管理。S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来禁用或启用 ACL。默认情况下，ACL 处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对数据的访问权限。
- 适用于 S3 的 IAM Access Analyzer - 评估和监控您的 S3 存储桶访问策略，确保这些策略仅提供对 S3 资源的预期访问权限。

管理和监控您的存储

- 管理存储 - 在 Amazon S3 中创建存储桶和上载对象后，您可以管理对象存储。例如，您可以使用 S3 版本控制和 S3 复制进行灾难恢复，使用 S3 生命周期管理存储成本，使用 S3 对象锁定来满足合规性要求。
- 监控存储 - 监控是保持 Amazon S3 和您的 AWS 解决方案的可靠性、可用性和性能的重要方面。您可以监控存储活动和成本。我们推荐您从 AWS 解决方案的所有方面收集监控数据，更轻松地调试出现的多点故障。
- 分析和见解 - 您可以使用 Amazon S3 中的分析和见解来了解、分析和优化存储使用情况。例如，使用 Amazon S3 Storage Lens 存储统计管理工具 了解、分析和优化存储。S3 Storage Lens 存储统计管理工具提供了超过 29 个使用和活动指标以及交互式仪表板，聚合整个组织、特定账户、区域、存储桶或前缀的数据。使用存储类分析来分析存储访问模式，决定何时需要将数据移动到更经济高效的存储类。

用 Amazon S3 进行开发

Amazon S3 是一项 REST 服务。您可以使用 REST API 或 AWS SDK 库（打包了底层 Amazon S3 REST API 以简化编程任务），向 Amazon S3 发送请求。您也可以使用 AWS Command Line Interface (AWS CLI) 调用Amazon S3 API。有关更多信息，请参阅 提出请求。

Amazon S3 REST API 是面向 Amazon S3 的 HTTP 接口。借助 REST API，您可以使用标准的 HTTP 请求创建、提取和删除存储桶和对象。要使用 REST API，您可以借助任何支持 HTTP 的工具包。只要对象是匿名可读的，您甚至可以使用浏览器来提取它们。有关更多信息，请参阅 使用 REST API 进行 Amazon S3 开发。

我们提供以下资源来帮助您使用所选的语言创建应用程序：

AWS CLI

您可以使用 AWS CLI 访问 Amazon S3 功能。要下载并配置 AWS CLI，请参阅 使用 AWS CLI 进行 Amazon S3 开发。

AWS CLI 提供两个层级的命令来访问 Amazon S3：高级别 (s3) 命令和 API 级别（ s3api 和 s3control ）命令。高级 S3 命令构成简化了常见任务的执行，如创建、操作和删除对象及存储桶。s3api 和 s3control 命令提供对所有 Amazon S3 API 操作的直接访问，您可以使用它们执行单凭高级别命令无法完成的高级操作。

有关 Amazon S3 的列表 AWS CLI 命令，请参见 s3、s3api 和 s3control。

AWS SDK 和 Explorer

使用 Amazon S3 开发应用程序时，您可以使用 AWS SDK。AWS SDK 包装了底层 REST API，可以简化您的编程任务。还提供 AWS Mobile SDK 和 Amplify JavaScript 库，用于使用 AWS 构建互连移动和 Web 应用程序。

除了 AWS SDK 外，AWS Explorer 也适用于 Visual Studio 和 Eclipse for Java IDE。在此情况下，可以将 SDK 和 Explorer 捆绑在一起作为 AWS 工具包。

有关更多信息，请参阅 使用 AWS SDK 通过 Amazon S3 进行开发。

示例代码和库

AWS 开发人员中心和 AWS 示例代码目录会特别针对 Amazon S3 编写示例代码和库。您可以运用这些代码示例了解 Amazon S3 API 的实施。您也可以查看 Amazon 简单存储服务 API 参考，了解 Amazon S3 API 操作的详细信息。

从教程中学习

开始使用分步教程，了解有关 Amazon S3 的更多信息。这些教程适用于实验室类型环境，使用了虚构的公司名称、用户名称等。其目的在于提供一般性指导。在未进行仔细审核并进行改进以满足组织环境的独特需求的情况下，这些过程不适合在您的生产环境中直接使用。

开始使用

- 教程：使用 Amazon S3 存储和检索文件
- 教程：开始使用 S3 Intelligent-Tiering

- 教程：开始使用 Amazon S3 Glacier 存储类

优化存储成本

- 教程：开始使用 S3 Intelligent-Tiering
- 教程：开始使用 Amazon S3 Glacier 存储类
- 教程：使用 S3 Storage Lens 存储统计管理工具优化成本和了解使用情况

管理存储

- 教程：Amazon S3 多区域接入点入门
- 教程：使用 S3 批量复制来复制 Amazon S3 存储桶中的现有对象

托管视频和网站

- 教程：使用 Amazon S3、Amazon CloudFront 和 Amazon Route 53 托管点播流视频
- 教程：在 Amazon S3 上配置静态网站
- 教程：使用注册到 Route 53 的自定义域配置静态网站

处理数据

- 教程：使用 S3 对象 Lambda 转换应用程序的数据
- 教程：使用 S3 对象 Lambda 和 Amazon Comprehend 检测和修订 PII 数据
- 教程：使用 S3 对象 Lambda 在检索图像时对其动态加水印
- 教程：使用 S3 批量操作、AWS Lambda 和 AWS Elemental MediaConvert 对视频进行批量转码

保护数据

- 教程：使用额外的校验和检查 Amazon S3 中数据的完整性
- 教程：使用 S3 复制在 AWS 区域内部以及之间复制数据
- 教程：使用 S3 版本控制、S3 对象锁定和 S3 复制保护 Amazon S3 上的数据免遭意外删除或应用程序错误
- 教程：使用 S3 批量复制来复制 Amazon S3 存储桶中的现有对象

探索培训和支持

您可以向 AWS 专家学习，提高您的技能并获得专家协助，实现您的目标。

- 培训 - 培训资源提供了学习 Amazon S3 的实践方法。有关更多信息，请参阅 AWS 培训和认证和 AWS 线上技术讲座。
- 论坛 - 您可以在论坛中查看帖子，了解使用 Amazon S3 的优势和局限。您也可以发布问题。有关更多信息，请参阅论坛。
- 技术支持 - 如果有其他问题，请联系技术支持。

教程

以下教程提供了 Amazon S3 的常见任务的完整端到端过程。这些教程适用于实验室类型环境，使用了虚构的公司名称、用户名称等。其目的在于提供一般性指导。在未进行仔细审核并进行改进以满足组织环境的独特需求的情况下，这些过程不适合在您的生产环境中直接使用。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

开始使用

- 教程：使用 Amazon S3 存储和检索文件
- 教程：开始使用 S3 Intelligent-Tiering
- 教程：开始使用 Amazon S3 Glacier 存储类

优化存储成本

- 教程：开始使用 S3 Intelligent-Tiering
- 教程：开始使用 Amazon S3 Glacier 存储类
- 教程：使用 S3 Storage Lens 存储统计管理工具优化成本和了解使用情况

管理存储

- 教程：Amazon S3 多区域接入点入门
- 教程：使用 S3 批量复制来复制 Amazon S3 存储桶中的现有对象

托管视频和网站

- 教程：使用 Amazon S3、Amazon CloudFront 和 Amazon Route 53 托管点播流视频
- 教程：在 Amazon S3 上配置静态网站

- 教程：使用注册到 Route 53 的自定义域配置静态网站

处理数据

- 教程：使用 S3 对象 Lambda 转换应用程序的数据
- 教程：使用 S3 对象 Lambda 和 Amazon Comprehend 检测和修订 PII 数据
- 教程：使用 S3 对象 Lambda 在检索图像时对其动态加水印
- 教程：使用 S3 批量操作、AWS Lambda 和 AWS Elemental MediaConvert 对视频进行批量转码

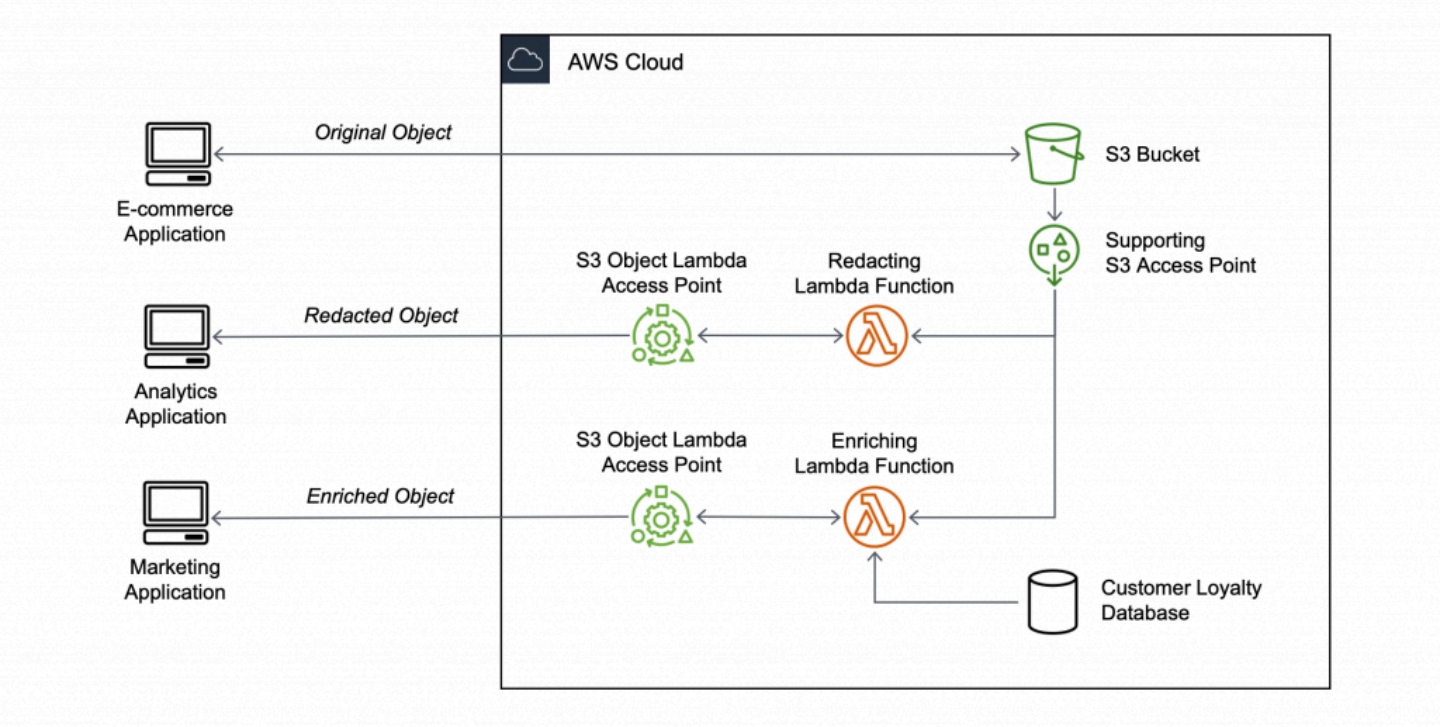
保护数据

- 教程：使用额外的校验和检查 Amazon S3 中数据的完整性
- 教程：使用 S3 复制在 AWS 区域内和区域之间复制数据
- 教程：使用 S3 版本控制、S3 对象锁定和 S3 复制保护 Amazon S3 上的数据免遭意外删除或应用程序错误
- 教程：使用 S3 批量复制来复制 Amazon S3 存储桶中的现有对象

教程：使用 S3 对象 Lambda 转换应用程序的数据

在 Amazon S3 中存储数据时，您可以轻松共享数据供多个应用程序使用。但是，每个应用程序可能具有独特的数据格式要求，可能需要针对特定用例修改或处理您的数据。例如，由电子商务应用程序创建的数据集可能包括个人识别信息（PII）。处理相同的数据以进行分析时，不需要此 PII，应该进行修订。但是，如果将同一数据集用于营销活动，可能需要使用其他详细信息来丰富数据，例如来自客户忠诚度数据库的信息。

采用 S3 对象 Lambda，可以添加自己的代码处理从 S3 检索的数据，然后再将其返回应用程序。具体来说，您可以配置 AWS Lambda 函数并将其附加到 S3 对象 Lambda 接入点。当应用程序通过 S3 对象 Lambda 接入点发送标准 S3 GET 请求时，将调用指定的 Lambda 函数，以处理通过支持的 S3 接入点从 S3 存储桶检索到的任何数据。然后，S3 对象 Lambda 接入点将转换后的结果返回给应用程序。您可以创作和执行自己的自定义 Lambda 函数，根据您的特定用例定制 S3 对象 Lambda 数据转换，所有这些都不需要更改您的应用程序。



目标

在本教程中，您将了解如何将自定义代码添加到标准 S3 GET 请求中，修改从 S3 检索到的请求对象，从而使该对象适合请求客户端或应用程序的需要。具体而言，您将学习如何通过 S3 对象 Lambda 将存储在 S3 中的原始对象中所有文本转换为大写。

> **Note**
>
> 本教程使用 Python 代码来转换数据，有关使用其它 AWS SDK 的示例，请参阅 AWS SDK 代码示例库中的 Transform data for your application with S3 Object Lambda。

主题

先决条件

在开始本教程之前，您必须具有 AWS 账户，您可使用具有正确权限的 AWS Identity and Access Management（IAM）用户登录此账户。您还必须安装 Python 3.8 或更高版本。

分步

- 在您的 AWS 账户 中创建具有权限的 IAM 用户（控制台）
- 在本地计算机上安装 Python 3.8 或更高版本

在您的 AWS 账户 中创建具有权限的 IAM 用户（控制台）

您可以为此教程创建 IAM 用户。要完成本教程，您的 IAM 用户必须附加以下 IAM 策略才能访问相关 AWS 资源并执行特定操作。有关如何创建 IAM 用户的更多信息，请参阅 《IAM 用户指南》中的创建 IAM 用户（控制台）。

您的 IAM 用户需要以下策略：

- AmazonS3FullAccess - 授予对所有 Amazon S3 操作的权限，包括创建和使用对象 Lambda 接入点的权限。
- AWSLambda_FullAccess - 对所有 Lambda 操作授予权限。
- IAMFullAccess - 对所有 IAM 操作授予权限。
- IAMAccessAnalyzerReadOnlyAccess - 授予读取 IAM Access Analyzer 提供的所有访问信息的权限。
- CloudWatchLogsFullAccess – 授予对 CloudWatch Logs 的完全访问权限。

> **Note**
>
> 为简单起见，本教程创建并使用 IAM 用户。完成本教程后，请记住 删除 IAM 用户。对于生产用途，我们建议您遵循《IAM 用户指南》中的 IAM 中的安全最佳实践。最佳实践要求人类用户将联合身份验证与身份提供商结合使用，以使用临时凭证访问 AWS。另一项最佳实践是要求

工作负载使用带有 IAM 角色的临时凭证访问 AWS。要了解如何使用 AWS IAM Identity Center 创建具有临时凭证的用户，请参阅《AWS IAM Identity Center 用户指南》中的入门。
本教程还使用完全访问 AWS 托管式策略。对于生产用途，建议您根据安全最佳实践仅授予用例所需的最低权限。

在本地计算机上安装 Python 3.8 或更高版本

使用以下过程在本地计算机上安装 Python 3.8 或更高版本。有关安装说明的更多信息，请参阅《Python 初学者指南》中的下载 Python 页面。

1. 打开本地终端或 shell 并运行以下命令以确定是否已安装 Python，如果已安装，确定已安装的版本。

   ```
   python --version
   ```

2. 如果没有 Python 3.8 或更高版本，则下载适合您的本地计算机的官方安装。
3. 双击下载的文件运行安装程序，然后按照步骤完成安装。

   对于 Windows 用户，在安装向导中选择将 Python 3.X 添加到 PATH，然后选择立即安装。
4. 通过关闭并重新打开终端来重新启动它。
5. 运行以下命令以验证是否正确安装了 Python 3.8或更高版本。

   对于 macOS 用户，运行以下命令：

   ```
   python3 --version
   ```

   对于 Windows 用户，运行此命令：

   ```
   python --version
   ```

6. 运行以下命令验证是否安装了 pip3 包管理器。如果您在命令响应中看到 pip 版本号和 python 3.8 或更高版本，则意味着 pi3 软件包管理器已成功安装。

   ```
   pip --version
   ```

步骤 1：创建 S3 存储桶

创建存储桶来存储您计划转换的原始数据。

创建存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择创建存储桶。

   此时将打开创建存储桶页面。
4. 对于存储桶名称，输入您的存储桶名称（例如，**tutorial-bucket**）。

   有关 Amazon S3 存储桶命名的更多信息，请参阅 存储桶命名规则。
5. 对于区域，选择要放置存储桶的 AWS 区域。

   有关存储桶区域的更多信息，请参阅 存储桶概述。
6. 对于此存储桶的屏蔽公共访问权限设置，保留默认值（屏蔽所有公共访问权限已启用）。

   除非您需要为您的用例关闭一个或多个屏蔽公共访问权限设置，否则建议您将所有这些设置保持为启用状态。有关屏蔽公共访问权限的更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。
7. 对于其余设置，保留默认值。

   （可选）如果要为您的特定用例配置其他存储桶设置，请参阅 创建存储桶。
8. 选择创建存储桶。

步骤 2：将文件上传到 S3 存储桶。

将文本文件上传到 S3 存储桶。此文本文件包含本教程稍后将转换为大写的原始数据。

例如，您可以上传包含以下文本的 `tutorial.txt`文件：

```
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.
```

要上传文件到存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择您在步骤 1中创建的存储桶名称（例如，**tutorial-bucket**）上传您的文件。
4. 在存储桶的对象 选项卡上，选择上传。
5. 在上传页面的文件和文件夹下，选择添加文件。
6. 选择要上传的文件，然后选择打开。例如，您可以上传之前提到的 `tutorial.txt` 文件示例。
7. 选择上传。

步骤 3：创建 S3 接入点

要使用 S3 对象 Lambda 接入点访问和转换原始数据，您必须创建一个 S3 接入点，并将其与您在步骤 1 中创建的 S3 存储桶关联。接入点必须与要变换的对象位于同一 AWS 区域 位置。

在本教程的后面部分，您将使用此接入点作为对象 Lambda 接入点的支持接入点。

创建接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择接入点。
3. 在接入点页面上，选择创建接入点。
4. 在接入点名称字段中输入接入点名称（例如，**tutorial-access-point**）。

   有关命名接入点的更多信息，请参阅命名 Amazon S3 接入点的规则。
5. 在存储桶名称字段中，输入您在步骤 1中创建的存储桶名称（例如，**tutorial-bucket**)。S3 将接入点附加到存储桶。

   您也可以选择浏览 S3 浏览和搜索账户中的存储桶。如果选择浏览 S3，请选择所需的存储桶，然后选择选择路径使用该存储桶的名称填充存储桶名称字段。
6. 适用于网络源中，选择互联网。

   有关接入点的网络源的更多信息，请参阅创建限制到 Virtual Private Cloud 的接入点。

7. 默认情况下，为您的接入点启用所有屏蔽公共访问权限设置。我们建议您将屏蔽所有 公共访问权限保留为启用状态。

   有关更多信息，请参阅 管理接入点的公有访问。

8. 对于所有其他接入点设置，保留默认设置。

   （可选）您可以修改接入点设置以支持您的用例。在本教程中，我们建议您保留默认设置。

   （可选）如果需要管理对接入点的访问，可以指定接入点策略。有关更多信息，请参阅 接入点策略示例。

9. 选择 Create access point（创建接入点）。

步骤 4：创建 Lambda 函数

要转换原始数据，请创建 Lambda 函数以与 S3 对象 Lambda 接入点结合使用。

分步

- 编写 Lambda 函数代码并使用虚拟环境创建部署包
- 使用执行角色创建 Lambda 函数（控制台）
- 用 .zip 文件归档部署 Lambda 函数代码，并配置 Lambda 函数（控制台）

编写 Lambda 函数代码并使用虚拟环境创建部署包

1. 在本地计算机上，使用文件夹名称 `object-lambda` 创建一个文件夹，供本教程稍后使用。
2. 在 `object-lambda` 文件夹中，使用 Lambda 函数创建一个文件，该函数将原始对象中的所有文本更改为大写。例如，您可以使用 Python 编写的以下函数。将此函数保存在名为的文件 `transform.py` 中。

```
import boto3
import requests
from botocore.config import Config

# This function capitalizes all text in the original object
def lambda_handler(event, context):
    object_context = event["getObjectContext"]
    # Get the presigned URL to fetch the requested original object
    # from S3
    s3_url = object_context["inputS3Url"]
```

```
    # Extract the route and request token from the input context
    request_route = object_context["outputRoute"]
    request_token = object_context["outputToken"]

    # Get the original S3 object using the presigned URL
    response = requests.get(s3_url)
    original_object = response.content.decode("utf-8")

    # Transform all text in the original object to uppercase
    # You can replace it with your custom code based on your use case
    transformed_object = original_object.upper()

    # Write object back to S3 Object Lambda
    s3 = boto3.client('s3', config=Config(signature_version='s3v4'))
    # The WriteGetObjectResponse API sends the transformed data
    # back to S3 Object Lambda and then to the user
    s3.write_get_object_response(
        Body=transformed_object,
        RequestRoute=request_route,
        RequestToken=request_token)

    # Exit the Lambda function: return the status code
    return {'status_code': 200}
```

> **Note**
>
> 前面的示例 Lambda 函数将整个请求的对象加载到内存中，然后再将其转换为客户端。您还可以从 S3 流式传输对象，避免将整个对象加载到内存中。这种方法在处理大型对象时非常有用。有关使用对象 Lambda 接入点流式处理响应的详细信息，请参阅在 Lambda 中处理 GetObject 请求中的流式处理示例。

编写 Lambda 函数以用于 S3 对象 Lambda 接入点时，该函数基于 S3 对象 Lambda 提供给 Lambda 函数的输入事件上下文。事件上下文提供有关在从 S3 对象 Lambda 传递到 Lambda 的事件中发出请求的信息。它包含用于创建 Lambda 函数的参数。

用于创建前面的 Lambda 函数的字段如下所示：

字段 `getObjectContext` 表示与 Amazon S3 和 S3 对象 Lambda连接的输入和输出详细信息。它具有以下字段：

- `inputS3Url` - Lambda 函数可用于从支持接入点下载原始对象的预签名 URL。通过使用预签名 URL，Lambda 函数无需具有 Amazon S3 读取权限即可检索原始对象，只能访问每次调用处理的对象。
- `outputRoute` - 当 Lambda 函数调用 `WriteGetObjectResponse` 以发回转换后的对象时，添加到 S3 对象 Lambda URL 的路由令牌。
- `outputToken` - S3 对象 Lambda 在发送回转换对象时用于将 `WriteGetObjectResponse` 调用与原始调用方匹配的令牌。

有关事件上下文中所有字段的更多信息，请参阅 事件上下文格式和用法 和 为 S3 对象 Lambda 接入点编写 Lambda 函数。

3. 在您的本地终端上，输入以下命令来安装 `virtualenv` 包：

```
python -m pip install virtualenv
```

4. 在您的本地终端中，打开之前创建的 `object-lambda` 文件夹，然后输入以下命令创建并初始化名为的虚拟环境 `venv`。

```
python -m virtualenv venv
```

5. 要激活虚拟环境，请输入以下命令，以执行环境的文件夹中的 `activate` 文件：

对于 macOS 用户，运行以下命令：

```
source venv/bin/activate
```

对于 Windows 用户， 运行此命令：

```
.\venv\Scripts\activate
```

现在，命令提示符更改为显示 (venv)，表示虚拟环境处于活动状态。

6. 要安装所需的库，请在 `venv` 虚拟环境中运行下列命令行。

这些命令会安装您 `lambda_handler` Lambda 函数依赖关系的更新版本。这些依赖关系是 AWS SDK for Python (Boto3) 和请求模块。

```
pip3 install boto3
```

```
pip3 install requests
```

7. 要停用虚拟环境，运行以下命令：

```
deactivate
```

8. 要在 object-lambda 目录的根目录下，将已安装的库作为 .zip 文件（文件名为 lambda.zip）来创建部署程序包，请在本地终端中逐行运行以下命令。

> **Tip**
>
> 可能需要调整以下命令才能在您的特定环境中工作。例如，库可能出现在 site-packages 或 dist-packages 中，第一个文件夹可能是 lib 或者 lib64。另外，python 文件夹可能会使用不同的 Python 版本命名。您可以使用 pip show 命令定位特定包。

适用于 macOS 用户，运行以下命令：

```
cd venv/lib/python3.8/site-packages
```

```
zip -r ../../../../lambda.zip .
```

对于 Windows 用户，运行以下命令：

```
cd .\venv\Lib\site-packages\
```

```
powershell Compress-Archive * ../../../lambda.zip
```

最后一个命令将部署程序包保存到 object-lambda 目录的根目录中。

9. 将函数代码文件 transform.py 添加到部署程序包的根目录中。

适用于 macOS 用户，运行以下命令：

```
cd ../../../../
```

```
zip -g lambda.zip transform.py
```

对于 Windows 用户，运行以下命令：

```
cd ..\..\..\
```

```
powershell Compress-Archive -update transform.py lambda.zip
```

完成此步骤后，文件目录结构应如下：

```
lambda.zip$
  # transform.py
  # __pycache__
  | boto3/
  # certifi/
  # pip/
  # requests/
  ...
```

使用执行角色创建 Lambda 函数（控制台）

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 在左侧导航窗格中，选择函数。
3. 选择 Create function (创建函数)。
4. 选择从头开始创作。
5. 在基本信息中，执行以下操作：
   a. 对于 Function name（函数名称），请输入 **tutorial-object-lambda-function**。
   b. 适用于运行时间中，选择 Python 3.8 或更高版本。
6. 展开更改默认执行角色部分。在执行角色中，选择创建具有基本 Lambda 权限的新角色。

   在步骤 5 中，在本教程的后面，您将 AmazonS3ObjectLambdaExecutionRolePolicy 附加到 Lambda 函数的执行角色上。
7. 对于其余设置，保留默认值。

8. 选择 Create function（创建函数）。

用 .zip 文件归档部署 Lambda 函数代码，并配置 Lambda 函数（控制台）

1. 在 AWS Lambda 控制台位于 https://console.aws.amazon.com/lambda/ 中，选择左侧导航窗格中的 Functions（函数）。
2. 选择您之前创建的 Lambda 函数（例如 **`tutorial-object-lambda-function`**）。
3. 在 Lambda 函数的详细信息页面上，选择代码选项卡。在代码源部分，选择上传，然后选择 .zip 文件。
4. 选择上传以选择本地 `.zip` 文件。
5. 选择您之前创建的 `lambda.zip` 文件，然后选择打开。
6. 选择保存。
7. 在运行时间设置部分，选择编辑。
8. 在存储库的编辑运行时设置页面上，确认运行时设置为 Python 3.8 或更高版本。
9. 要告诉 Lambda 运行时要调用 Lambda 函数代码中的哪个处理程序方法，请输入 **`transform.lambda_handler`** 的处理程序。

   当您在 Python 中配置函数时，处理程序设置的值是文件名，也是处理程序模块的名称（由点分隔）。例如，`transform.lambda_handler` 调用在 `transform.py` 文件中定义的 `lambda_handler` 方法。
10. 选择保存。
11. （可选）在 Lambda 函数的详细信息页面上，选择配置选项卡。在左侧导航窗格中，选择常规配置，然后选择编辑。在超时字段中，输入 **`1`** 分钟 **`0`** 秒。对于其余设置，保留默认值，然后选择保存。

    超时 –是 Lambda 允许函数在停止调用之前为调用运行的时长。默认值为 3 秒。S3 对象 Lambda 使用的 Lambda 函数的最大持续时间为 60 秒。定价基于配置的内存量和代码运行的时间量。

步骤 5：为 Lambda 函数的执行角色配置 IAM 策略

要使 Lambda 函数能够提供自定义的数据和响应标头 `GetObject` 调用者，Lambda 函数的执行角色必须具有 IAM 权限才能调用 `WriteGetObjectResponse` API。

将 IAM 策略附加到 Lambda 函数角色

1. 在 AWS Lambda 控制台位于 https://console.aws.amazon.com/lambda/ 中，选择左侧导航窗格中的 Functions（函数）。
2. 选择您在步骤 4 中创建的函数（例如，**tutorial-object-lambda-function**）。
3. 在 Lambda 函数的详细信息页面上，选择配置选项卡，然后选择Permissions（权限）在左侧导航窗格中。
4. Unts执行角色中，选择 Role name（角色名称）。打开 IAM 控制台。
5. 在 IAM 控制台上对应于 Lambda 函数的执行角色的 Summary（摘要）页面上，选择 Permissions（权限）选项卡。然后，从 Add Permissions（添加权限）菜单中选择 Attach policies（附加策略）。
6. 在附加权限策略页的搜索字段中输入 **AmazonS3ObjectLambdaExecutionRolePolicy** 策略的名称。选中 AmazonS3ObjectLambdaExecutionRolePolicy 政策名称旁边的复选框。
7. 选择附加策略。

步骤 6：创建 S3 对象 Lambda 接入点

S3 对象 Lambda 接入点提供了直接从 S3 GET 请求调用 Lambda 函数的灵活性，以便该函数可以处理从 S3 接入点检索的数据。创建和配置 S3 对象 Lambda 接入点时，必须指定 Lambda 函数以调用并提供 JSON 格式的事件上下文作为 Lambda 要使用的自定义参数。

创建 S3 对象 Lambda 接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Object Lambda 接入点。
3. 在对象 Lambda 接入点页面上，选择创建对象 Lambda 接入点。
4. 对于对象 Lambda 接入点名称，输入想要用于对象 Lambda 接入点的名称（例如 **tutorial-object-lambda-accesspoint**）。
5. 适用于支持接入点中，输入或浏览到您在步骤 3（例如，**tutorial-access-point**），然后选择选择支持的接入点。
6. 对于 S3 APIs（S3 API），要从 S3 存储桶中检索对象以便 Lambda 函数进行处理，请选择 GetObject。
7. 对于调用 Lambda 函数，您可以为本教程选择以下两个选项之一。
   - 选择从您的账户中的函数选择，然后在 Lambda 函数下拉列表选择您在步骤 4中创建的 Lambda 函数（例如，**tutorial-object-lambda-function**）。

- 选择输入 ARN，然后输入您在步骤 4 中创建的 Lambda 函数的 Amazon 资源名称（ARN）。

8. 适用于Lambda 函数版本，选择$LATEST（您在步骤 4中创建的最新版 Lambda 函数)。
9. （可选）如果您需要 Lambda 函数识别和处理带有范围和分段编号标头的 GET 请求，请选择 Lambda 函数支持使用范围的请求和 Lambda 函数支持使用零件编号的请求。否则，请清除这两个复选框。

   有关如何将范围或分段编号与 S3 对象 Lambda 结合使用的更多信息，请参阅 使用 Range 和 partNumber 标头。
10. （可选）在 Payload（负载）- 可选下，添加 JSON 文本，向 Lambda 函数提供其他信息。

    负载是可选 JSON 文本，您可以将其作为来自特定 S3 对象 Lambda 接入点的所有调用的输入提供给 Lambda 函数。要自定义调用同一 Lambda 函数的多个对象 Lambda 接入点的行为，您可以使用不同的参数配置负载，从而扩展 Lambda 函数的灵活性。

    更多有关路径模式的信息，请参阅 事件上下文格式和用法。
11. （可选）对于请求指标 - 可选，选择禁用或启用以将 Amazon S3 监控添加到对象 Lambda 接入点。请求指标按标准 Amazon CloudWatch 费率计费。有关更多信息，请参阅 CloudWatch 定价。
12. 在对象 Lambda 接入点策略 - 可选下，保留默认设置。

    （可选）您可以设置资源策略。此资源策略授予 `GetObject` API 权限，以便使用指定的对象 Lambda 接入点。
13. 对于其余设置，保留默认值，然后选择创建对象 Lambda 接入点。

步骤 7：查看转换的数据

现在，S3 对象 Lambda 已准备好为您的使用案例转换您的数据。在本教程中，S3 对象 Lambda 将对象中的所有文本转换为大写。

分步

- 在 S3 对象 Lambda 接入点中查看转换后的数据
- 运行 Python 脚本以打印原始数据和转换数据

在 S3 对象 Lambda 接入点中查看转换后的数据

当您请求通过 S3 对象 Lambda 接入点检索文件时，您需要对 S3 对象 Lambda 进行 `GetObject` API 调用。S3 对象 Lambda 调用 Lambda 函数来转换您的数据，然后返回转换后的数据作为对标准 S3 的响应 `GetObject` API 调用。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择对象 Lambda 接入点。
3. 在对象 Lambda 接入点页面上，选择您在步骤 6 中创建的 S3 对象 Lambda 接入点（例如，**`tutorial-object-lambda-accesspoint`**）。
4. 在 S3 对象 Lambda 接入点的对象选项卡中，选择具有相同名称的文件（例如 `tutorial.txt`）作为您在步骤 2 中上传到 S3 存储桶的文件。

   此文件应包含所有转换的数据。
5. 要查看转换后的数据，请选择打开或者下载。

运行 Python 脚本以打印原始数据和转换数据

您可以将 S3 对象 Lambda 与现有应用程序结合使用。为此，请更新应用程序配置，以使用在步骤 6 中创建的新 S3 对象 Lambda 接入点 ARN 从 S3 检索数据。

以下示例 Python 脚本既打印来自 S3 存储桶的原始数据，也打印来自 S3 对象 Lambda 接入点的转换数据。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择对象 Lambda 接入点。
3. 在对象 Lambda 接入点页面上，选择在步骤 6 中创建的 S3 对象 Lambda 接入点左侧的单选按钮（例如，**`tutorial-object-lambda-accesspoint`**）。
4. 选择复制 ARN。
5. 保存 ARN 以供稍后使用。
6. 在本地计算机上编写 Python 脚本，以从 S3 存储桶中打印原始数据（例如，`tutorial.txt`）并从 S3 对象 Lambda 接入点中打印转换后的数据（例如 `tutorial.txt`）。您可以使用以下示例脚本。

```
import boto3
from botocore.config import Config

s3 = boto3.client('s3', config=Config(signature_version='s3v4'))

def getObject(bucket, key):
    objectBody = s3.get_object(Bucket = bucket, Key = key)
    print(objectBody["Body"].read().decode("utf-8"))
    print("\n")

print('Original object from the S3 bucket:')
# Replace the two input parameters of getObject() below with
# the S3 bucket name that you created in Step 1 and
# the name of the file that you uploaded to the S3 bucket in Step 2
getObject("tutorial-bucket",
          "tutorial.txt")

print('Object transformed by S3 Object Lambda:')
# Replace the two input parameters of getObject() below with
# the ARN of your S3 Object Lambda Access Point that you saved earlier and
# the name of the file with the transformed data (which in this case is
# the same as the name of the file that you uploaded to the S3 bucket
# in Step 2)
getObject("arn:aws:s3-object-lambda:us-west-2:111122223333:accesspoint/tutorial-
object-lambda-accesspoint",
          "tutorial.txt")
```

7. 使用自定义名称将 Python 脚本（例如，`tutorial_print.py`）保存您在步骤 4中本地计算机上中创建的文件夹中（例如，`object-lambda`）。
8. 在您的本地终端上，在步骤 4中创建的根目录下运行以下命令（例如，`object-lambda`）。

```
python3 tutorial_print.py
```

您应该通过终端同时看到原始数据和转换后的数据（所有文本均为大写）。如，您应看到类似于以下的文本。

```
Original object from the S3 bucket:
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.
```

```
Object transformed by S3 Object Lambda:
AMAZON S3 OBJECT LAMBDA TUTORIAL:
YOU CAN ADD YOUR OWN CODE TO PROCESS DATA RETRIEVED FROM S3 BEFORE
RETURNING IT TO AN APPLICATION.
```

第 8 步：清除

如果您仅出于练习目的通过 S3 对象 Lambda 创建静态网站，则删除您所分配的 AWS 资源，使其不再产生费用。

分步

- 删除对象 Lambda 接入点
- 删除 S3 接入点
- 删除您的 Lambda 函数的执行角色。
- 删除 Lambda 函数
- 删除 CloudWatch 日志组
- 删除 S3 源存储桶中的原始文件
- 删除 S3 源存储桶
- 删除 IAM 用户

删除对象 Lambda 接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择对象 Lambda 接入点。
3. 在对象 Lambda 接入点页面上，选择在步骤 6 中创建的 S3 对象 Lambda 接入点左侧的单选按钮（例如，**tutorial-object-lambda-accesspoint**）。
4. 选择删除。
5. 通过在显示的文本字段中输入对象 Lambda 接入点的名称，确认要删除的对象 Lambda 接入点，然后选择删除。

删除 S3 接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择接入点。
3. 导航到您在步骤 3创建的接入点（例如，**tutorial-access-point**），然后选择接入点名称旁的单选按钮。
4. 选择删除。
5. 在显示的文本字段中输入接入点名称，然后选择删除，确认您要删除接入点。

删除您的 Lambda 函数的执行角色。

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 在左侧导航窗格中，选择函数。
3. 选择您在步骤 4 中创建的函数（例如，**tutorial-object-lambda-function**）。
4. 在 Lambda 函数的详细信息页面上，选择配置选项卡，然后选择Permissions（权限）在左侧导航窗格中。
5. Unts执行角色中，选择 Role name（角色名称）。打开 IAM 控制台。
6. 在 Lambda 函数的执行角色 IAM 控制台的摘要页面上，选择删除角色。
7. 在 Delete role（删除角色）对话框中，选择 Yes, delete（是的，删除）。

删除 Lambda 函数

1. 在 AWS Lambda 控制台（位于 https://console.aws.amazon.com/lambda/）中，选择左侧导航窗格中的 Functions（函数）。
2. 选择您在步骤4中创建的函数名称左侧的复选框（例如，**tutorial-object-lambda-function**）。
3. 选择操作，然后选择删除。
4. 在删除函数对话框中，选择删除。

删除 CloudWatch 日志组

1. 访问 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。

2. 在左侧导航窗格中，选择日志组。
3. 查找名称以步骤4中创建的 Lambda 函数结尾的日志组（例如，**tutorial-object-lambda-function**）。
4. 选择日志组名称左侧的复选框。
5. 选择操作，然后选择删除日志组。
6. 在删除日志组对话框中，选择删除。

删除 S3 源存储桶中的原始文件

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶名称列表中，选择您在步骤 2中将原始文件上传到的存储桶名称（例如，**tutorial-bucket**）。
4. 选中要删除的对象名称左侧的复选框（例如，`tutorial.txt`）。
5. 选择删除。
6. 在永久删除对象？的删除对象页面上，在文本框中输入 **permanently delete** 确认您希望删除此对象。
7. 选择删除对象。

删除 S3 源存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择您在步骤 1中创建的存储桶名称旁的单选按钮（例如，**tutorial-bucket**）。
4. 选择删除。
5. 在删除存储桶页面上，通过在文本字段中输入存储桶名称来确认要删除存储桶，然后选择删除存储桶。

删除 IAM 用户

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择用户，然后选中要删除的用户旁的复选框。
3. 在页面的顶部，选择删除。
4. 在删除###?对话框中，在文本输入字段中输入用户名以确认删除用户。选择 Delete（删除）。

后续步骤

完成本教程后，您可以为您的用例自定义 Lambda 函数，以便修改标准 S3 GET 请求返回的数据。

以下是 S3 对象 Lambda 的常见用例列表：

- 屏蔽敏感数据以确保安全性和合规性。

  有关更多信息，请参阅 教程：使用 S3 对象 Lambda 和 Amazon Comprehend 检测和修订 PII 数据。
- 筛选某些数据行以提供特定信息。
- 使用来自其他服务或数据库的信息增加数据。
- 跨数据格式转换，例如将 XML 转换为 JSON 以实现应用程序兼容性。
- 在下载文件时压缩或解压缩文件。
- 调整图像大小和水印图像。

  有关更多信息，请参阅教程：使用 S3 对象 Lambda 在检索图像时对其动态加水印。
- 实施自定义授权规则以访问数据。

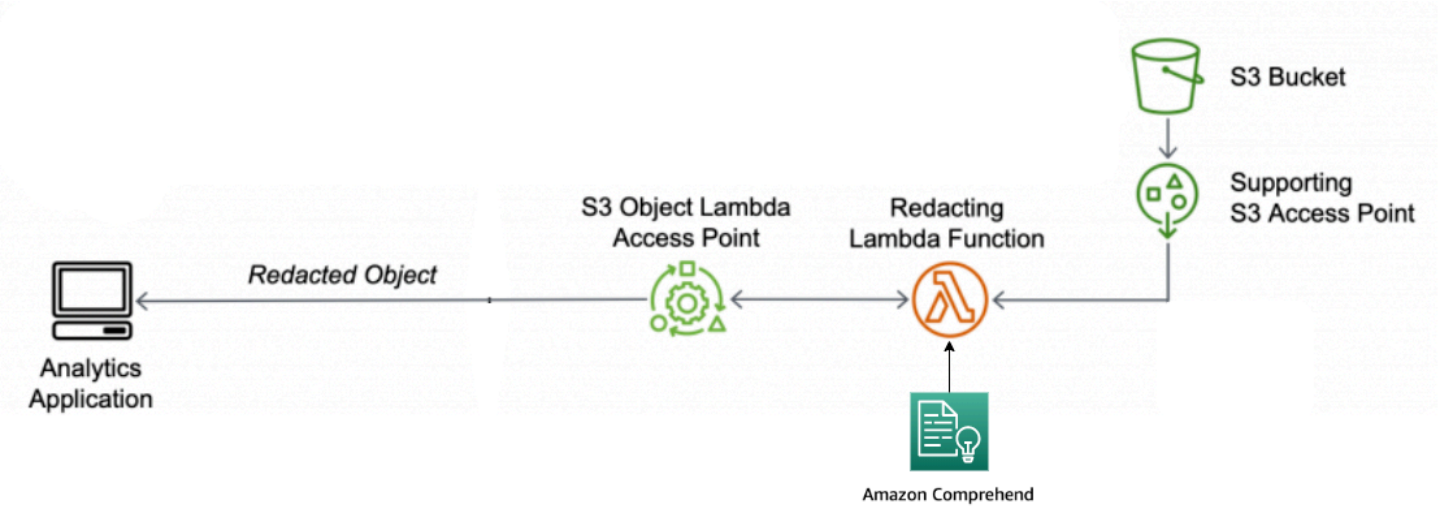
有关 S3 对象 Lambda 的更多信息，请参阅 使用 S3 对象 Lambda 转换对象。

教程：使用 S3 对象 Lambda 和 Amazon Comprehend 检测和修订 PII 数据

将 Amazon S3 用于多个应用程序和用户访问的共享数据集时，请务必将特权信息（如个人身份信息 (PII)）限制为仅授权实体。例如，当营销应用程序使用某些包含 PII 的数据时，可能需要首先掩盖 PII

数据以满足数据隐私要求。此外，分析应用程序使用生产订单清单数据集时，可能需要首先编辑客户信用卡信息，防止意外的数据泄露。

用 S3 对象 Lambda 和预构建的 AWS Lambda 函数，您可以在将从 S3 检索 Amazon Comprehend 的 PII 数据返回到应用程序之前进行保护。具体来说，您可以使用预构建的 Lambda 函数作为编校函数，并将其附加到 S3 对象 Lambda 接入点。应用程序（例如，分析应用程序）发送标准 S3 GET 请求时，这些通过 S3 对象 Lambda 接入点发出的请求将调用预构建的编校 Lambda 函数，以检测和编校通过支持的 S3 接入点从 S3 存储桶检索到的 PII 数据。然后，S3 对象 Lambda 接入点将编校的结果返回到应用程序。



在此过程中，无论 PII 在文本中的存在方式如何（例如，数字或单词和数字的组合），预构建的 Lambda 函数均使用 Amazon Comprehend，这是一种自然语言处理 (NLP) 服务，用于捕获 PII 表示方式的变化。Amazon Comprehend 甚至可以用文本中的上下文来了解 4 位数字是否是 PIN、社会保险号 (SSN) 的最后四个数字还是年份。Amazon Comprehend 处理 UTF-8 格式的任何文本文件，并可以大规模保护 PII，而不会影响准确性。有关更多信息，请参阅 《Amazon Comprehend 开发人员指南》中的什么是 Amazon Comprehend？

目标

在本教程中，您将学习如何将 S3 对象 Lambda 与预构建的 Lambda 函数一起使用 `ComprehendPiiRedactionS3ObjectLambda`。此函数使用 Amazon Comprehend 来检测 PII 实体。然后，通过用星号替换这些实体来修订它们。通过修订 PII，您可以隐藏敏感数据，这有助于提高安全性和合规性。

您还可以了解在 AWS Serverless Application Repository 中如何使用和配置预构建的 AWS Lambda 函数与 S3 对象 Lambda 一起使用便于部署。

主题

先决条件：创建具有权限的 IAM 用户

在开始本教程之前，您必须具有 AWS 账户，您可使用具有正确权限的 AWS Identity and Access Management 用户（IAM 用户）登录此账户。

您可以为此教程创建 IAM 用户。要完成本教程，您的 IAM 用户必须附加以下 IAM 策略才能访问相关 AWS 资源并执行特定操作。

> **Note**
>
> 为简单起见，本教程创建并使用 IAM 用户。完成本教程后，请记住 删除 IAM 用户。对于生产用途，我们建议您遵循《IAM 用户指南》中的 IAM 中的安全最佳实践。最佳实践要求人类用户将联合身份验证与身份提供商结合使用，以使用临时凭证访问 AWS。另一项最佳实践是要求工作负载使用带有 IAM 角色的临时凭证访问 AWS。要了解如何使用 AWS IAM Identity Center 创建具有临时凭证的用户，请参阅《AWS IAM Identity Center 用户指南》中的入门。
> 本教程还使用完全访问策略。对于生产用途，我们建议您根据安全最佳实践仅授予用例所需的最低权限。

您的 IAM 用户需要以下 AWS 托管策略：

- AmazonS3FullAccess - 授予对所有 Amazon S3 操作的权限，包括创建和使用对象 Lambda 接入点的权限。
- AWSLambda_FullAccess - 对所有 Lambda 操作授予权限。
- AWSCloudFormationFullAccess - 为所有 AWS CloudFormation 操作授予权限。

- IAMFullAccess - 对所有 IAM 操作授予权限。
- IAMAccessAnalyzerReadOnlyAccess - 授予读取 IAM Access Analyzer 提供的所有访问信息的权限。

您可以在创建 IAM 用户时直接附加这些现有策略。有关如何创建 IAM 用户的更多信息，请参阅 《IAM 用户指南》中的创建 IAM 用户（控制台）。

此外，您的 IAM 用户需要客户托管策略。将 IAM 用户权限授予所有 AWS Serverless Application Repository 资源和操作，您必须创建 IAM 策略并将此策略附加给 IAM 用户。

创建一个 IAM 策略并将其附加到 IAM 用户

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Policies（策略）。
3. 选择创建策略。
4. 在可视化编辑器选项卡上，服务，选择服务。然后，选择 Serverless Application Repository。
5. 对于操作，在手动操作中，选择此教程中所有 Serverless Application Repository 操作（serverlessrepo:*）。

   作为安全最佳实践，您应根据用例仅允许用户所需的操作和资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。
6. 对于资源中，选择本教程中的所有资源。

   作为最佳实践，您应仅为特定账户中的特定资源定义权限。您也可以使用条件键授予最低权限。有关更多信息，请参阅 《IAM 用户指南》 中的授予最低权限。
7. 选择下一步：标签。
8. 选择下一步：审核。
9. 在查看策略页面上，为创建的策略输入名称（例如，**`tutorial-serverless-application-repository`**）和描述（可选）。查看策略摘要以确保您授予了所需的权限，然后选择创建策略以保存新策略。
10. 在左侧导航窗格中，选择 用户。然后，为本教程选择 IAM 用户。
11. 在所选用户的摘要页面上，选择权限选项卡，然后选择添加权限。
12. 在 Grant permissions (授予权限) 下，选择 Attach existing policies directly (直接挂载现有策略)。

13. 选择刚创建的策略旁边的复选框（例如，**tutorial-serverless-application-repository**），然后选择下一步：审查。
14. 在策略摘要下，查看摘要确保附加了预期的策略。然后选择 添加权限。

步骤 1：创建 S3 存储桶

创建存储桶来存储您计划转换的原始数据。

创建存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择创建存储桶。

   此时将打开创建存储桶页面。
4. 对于存储桶名称，输入您的存储桶名称（例如，**tutorial-bucket**）。

   有关 Amazon S3 存储桶命名的更多信息，请参阅 存储桶命名规则。
5. 对于区域，选择要放置存储桶的 AWS 区域。

   有关存储桶区域的更多信息，请参阅 存储桶概述。
6. 对于此存储桶的屏蔽公共访问权限设置，保留默认值（屏蔽所有公共访问权限已启用）。

   除非您需要为您的用例关闭一个或多个屏蔽公共访问权限设置，否则建议您将所有这些设置保持为启用状态。有关屏蔽公共访问权限的更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。
7. 对于其余设置，保留默认值。

   （可选）如果要为您的特定用例配置其他存储桶设置，请参阅 创建存储桶。
8. 选择创建存储桶。

步骤 2：将文件上传到 S3 存储桶。

将包含各种类型的已知 PII 数据（如姓名、银行信息、电话号码和SSN）的文本文件作为原始数据上传到 S3 存储桶中，您将在本教程稍后部分对PII进行修订。

例如，您可以上传 `tutorial.txt` 文件。这是 Amazon Comprehend 的示例输入文件。

```
Hello Zhang Wei, I am John. Your AnyCompany Financial Services,
LLC credit card account 1111-0000-1111-0008 has a minimum payment
of $24.53 that is due by July 31st. Based on your autopay settings,
we will withdraw your payment on the due date from your
bank account number XXXXXX1111 with the routing number XXXXX0000.

Your latest statement was mailed to 100 Main Street, Any City,
WA 98121.
After your payment is received, you will receive a confirmation
text message at 206-555-0100.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at 206-555-0199 or
email at support@anycompany.com.
```

要上传文件到存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择您在步骤 1中创建的存储桶名称（例如，**tutorial-bucket**）上传您的文件。
4. 在存储桶的对象 选项卡上，选择上传。
5. 在上传页面的文件和文件夹下，选择添加文件。
6. 选择要上传的文件，然后选择打开。例如，您可以上传之前提到的 `tutorial.txt` 文件示例。
7. 选择上传。

步骤 3：创建 S3 接入点

要使用 S3 对象 Lambda 接入点访问和转换原始数据，您必须创建一个 S3 接入点，并将其与您在步骤 1 中创建的 S3 存储桶关联。接入点必须与要转换的对象位于同一 AWS 区域中。

在本教程的后面部分，您将使用此接入点作为对象 Lambda 接入点的支持接入点。

创建接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择接入点。

3. 在接入点页面上，选择创建接入点。
4. 在接入点名称字段中输入接入点名称（例如，**tutorial-pii-access-point**）。

   有关命名接入点的更多信息，请参阅命名 Amazon S3 接入点的规则。
5. 在存储桶名称字段中，输入您在步骤 1中创建的存储桶名称（例如，**tutorial-bucket**)。S3 将接入点附加到存储桶。

   您也可以选择浏览 S3 浏览和搜索账户中的存储桶。如果选择浏览 S3，请选择所需的存储桶，然后选择选择路径使用该存储桶的名称填充存储桶名称字段。
6. 适用于网络源中，选择互联网。

   有关接入点的网络源的更多信息，请参阅创建限制到 Virtual Private Cloud 的接入点。
7. 默认情况下，为接入点启用所有屏蔽公共访问权限设置。我们建议您将屏蔽所有 公共访问权限保留为启用状态。有关更多信息，请参阅 管理接入点的公有访问。
8. 对于所有其他接入点设置，保留默认设置。

   （可选）您可以修改接入点设置以支持您的用例。在本教程中，我们建议您保留默认设置。

   （可选）如果需要管理对接入点的访问，可以指定接入点策略。有关更多信息，请参阅 接入点策略示例。
9. 选择 Create access point（创建接入点）。

步骤 4：配置和部署预构建的 Lambda 函数

要编校 PII 数据，请配置和部署预构建的 AWS Lambda 函数 `ComprehendPiiRedactionS3ObjectLambda` 以便与 S3 对象 Lambda 接入点结合使用。

要配置和部署 Lambda 函数

1. 登录到 AWS Management Console，然后查看 AWS Serverless Application Repository 中的 `ComprehendPiiRedactionS3ObjectLambda` 函数。
2. 对于应用程序设置，在应用程序名称下，请保留本教程默认值 (`ComprehendPiiRedactionS3ObjectLambda`)。

   （可选）您可以输入要为此应用程序提供的名称。如果计划为同一共享数据集的不同访问需求配置多个 Lambda 函数，则可能需要执行此操作。
3. 对于 MaskCharacter，请保留默认值 (*)。掩码字符将替换已修订的 PII 实体中的每个字符。

4. 对于 MaskMode，请保留默认值 (MASK)。MaskMode 值指定 PII 实体是否使用 `MASK` 字符或 `PII_ENTITY_TYPE` 值。
5. 要修订数据指定类型，对于 PiiEntityTypes，请保留默认值 ALL。PiiEntityTypes 值指定要考虑修订的 PII 实体类型。

   有关支持的 PII 实体类型列表的更多信息，请参阅 《Amazon Comprehend 开发人员指南》中的检测个人身份信息 (PII)。
6. 对于其余设置，保留默认值。

   （可选）如果要为特定用例配置其他设置，请参阅页面左侧的自述文件部分。
7. 选中 I acknowledge that this app creates custom IAM roles (我确认此应用程序创建自定义 IAM 角色) 旁边的复选框。
8. 选择部署。
9. 在新应用程序的页面上，在资源下，选择Lambda 函数的逻辑 ID，在 Lambda 函数页面上部署的 Lambda 函数。

步骤 5：创建 S3 对象 Lambda 接入点

S3 对象 Lambda 接入点提供了直接从 S3 GET 请求调用 Lambda 函数的灵活性，以便该函数可以编校从 S3 接入点检索的 PII 数据。创建和配置 S3 对象 Lambda 接入点时，必须指定要调用的编校 Lambda 函数，并以 JSON 格式提供事件上下文作为供 Lambda 使用的自定义参数。

事件上下文提供有关在从 S3 对象 Lambda 传递到 Lambda 的事件中发出请求的信息。有关事件上下文中所有字段的更多信息，请参阅 事件上下文格式和用法。

创建 S3 对象 Lambda 接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Object Lambda 接入点。
3. 在对象 Lambda 接入点页面上，选择创建对象 Lambda 接入点。
4. 对于对象 Lambda 接入点名称，输入想要用于对象 Lambda 接入点的名称（例如 **`tutorial-pii-object-lambda-accesspoint`**）。
5. 适用于支持接入点中，输入或浏览到您在步骤 3（例如，**`tutorial-pii-access-point`**），然后选择选择支持的接入点。

6. 对于 S3 APIs（S3 API），要从 S3 存储桶中检索对象以便 Lambda 函数进行处理，请选择 GetObject。
7. 对于调用 Lambda 函数，您可以为本教程选择以下两个选项之一。
   - 选择从您的账户中的函数中选择，选择您在步骤 4中部署的来自 Lambda 函数下拉列表的 Lambda 函数（例如，**`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**）。
   - 选择输入 ARN，然后输入您在步骤 4 中创建的 Lambda 函数的 Amazon 资源名称（ARN）。
8. 对于 Lambda 函数版本中，选择$LATEST（您在步骤 4中部署的最新版本 Lambda 函数)。
9. （可选）如果您需要 Lambda 函数识别和处理带有范围和分段编号标头的 GET 请求，选择 Lambda 函数支持，使用范围的请求和 Lambda 函数支持带零件编号的请求。否则，请清除这两个复选框。

   有关如何将范围或分段编号与 S3 对象 Lambda 结合使用的更多信息，请参阅 使用 Range 和 partNumber 标头。
10. （可选）在 Payload（负载）- 可选下，添加 JSON 文本，向 Lambda 函数提供其他信息。

    负载是可选 JSON 文本，您可以将其作为来自特定 S3 对象 Lambda 接入点的所有调用的输入提供给 Lambda 函数。要自定义调用同一 Lambda 函数的多个对象 Lambda 接入点的行为，您可以使用不同的参数配置负载，从而扩展 Lambda 函数的灵活性。

    更多有关路径模式的信息，请参阅 事件上下文格式和用法。
11. （可选）对于请求指标 - 可选，选择禁用或启用以将 Amazon S3 监控添加到对象 Lambda 接入点。请求指标按标准 Amazon CloudWatch 费率计费。有关更多信息，请参阅 CloudWatch 定价。
12. 在对象 Lambda 接入点策略 - 可选下，保留默认设置。

    （可选）您可以设置资源策略。此资源策略授予 `GetObject` API 权限，以便使用指定的对象 Lambda 接入点。
13. 对于其余设置，保留默认值，选择创建对象 Lambda 接入点。

步骤 6：使用 S3 对象 Lambda 接入点检索编校的文件

现在，S3 对象 Lambda 已准备好从原始文件修订 PII 数据。

使用 S3 对象 Lambda 接入点检索编校的文件

当您请求通过 S3 对象 Lambda 接入点检索文件时，您需要对 S3 对象 Lambda 进行 `GetObject` API 调用。S3 对象 Lambda 调用 Lambda 函数来编辑 PII 数据，并将转换后的数据作为对标准 S3 `GetObject` API调用的响应返回。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择对象 Lambda 接入点。
3. 在对象 Lambda 接入点页面上，选择您在步骤 5 中创建的 S3 对象 Lambda 接入点（例如，**`tutorial-pii-object-lambda-accesspoint`**）。
4. 在 S3 对象 Lambda 接入点的对象选项卡中，选择具有相同名称的文件（例如 `tutorial.txt`）作为您在步骤 2 中上传到 S3 存储桶的文件。

   此文件应包含所有转换的数据。
5. 要查看转换后的数据，请选择打开或者下载。

   您应该能够看到修订的文件，如以下示例所示。

```
Hello *********. Your AnyCompany Financial Services,
LLC credit card account ******************* has a minimum payment
of $24.53 that is due by *********. Based on your autopay settings,
we will withdraw your payment on the due date from your
bank account ********** with the routing number *********.

Your latest statement was mailed to *********************************.
After your payment is received, you will receive a confirmation
text message at ************.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at ************ or
email at **********************.
```

步骤 7：清除

如果您仅作为学习练习通过 S3 对象 Lambda 修订数据，请删除分配的 AWS 资源，以便不再累积费用。

分步

- 删除对象 Lambda 接入点
- 删除 S3 接入点
- 删除 Lambda 函数
- 删除 CloudWatch 日志组
- 删除 S3 源存储桶中的原始文件
- 删除 S3 源存储桶
- 删除您的 Lambda 函数的 IAM 角色
- 删除 IAM 用户的客户托管策略
- 删除 IAM 用户

删除对象 Lambda 接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择对象 Lambda 接入点。
3. 在对象 Lambda 接入点页面上，选择在步骤 5 中创建的 S3 对象 Lambda 接入点（例如，**`tutorial-pii-object-lambda-accesspoint`**）左侧的选项按钮。
4. 选择删除。
5. 通过在显示的文本字段中输入对象 Lambda 接入点的名称，确认要删除的对象 Lambda 接入点，然后选择删除。

删除 S3 接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择接入点。
3. 导航到您在步骤 3中创建的接入点（例如，**`tutorial-pii-access-point`**），然后选择接入点名称旁边的选项按钮。
4. 选择删除。
5. 在显示的文本字段中输入接入点名称，然后选择删除，确认您要删除接入点。

删除 Lambda 函数

1. 在 AWS Lambda 控制台左侧导航窗格中 https://console.aws.amazon.com/lambda/ 中，选择函数。
2. 选择您在步骤 4 中创建的函数（例如，**serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**）。
3. 选择操作，然后选择删除。
4. 在删除函数对话框中，选择删除。

删除 CloudWatch 日志组

1. 访问 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择日志组。
3. 查找名称以步骤4中创建的 Lambda 函数结尾的日志组（例如，**serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**）。
4. 选择操作，然后选择删除日志组。
5. 在删除日志组对话框中，选择删除。

删除 S3 源存储桶中的原始文件

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶名称列表中，选择您在步骤 2中将原始文件上传到的存储桶名称（例如，**tutorial-bucket**)。
4. 选中要删除的对象名称左侧的复选框（例如，`tutorial.txt`）。
5. 选择删除。
6. 在永久删除对象？的删除对象页面上，在文本框中输入 **permanently delete** 确认您希望删除此对象。
7. 选择删除对象。

删除 S3 源存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择您在步骤 1中创建存储桶名称旁边的选项按钮（例如，**tutorial-bucket**）。
4. 选择删除。
5. 在删除存储桶页面上，通过在文本字段中输入存储桶名称来确认要删除存储桶，然后选择删除存储桶。

删除您的 Lambda 函数的 IAM 角色

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择角色，然后选中要删除的角色旁的复选框。角色名称以您在步骤 4中部署的 Lambda 函数名称开头（例如，**serverlessrepo-ComprehendPiiRedactionS3ObjectLambda**）。
3. 选择删除。
4. 在删除对话框中，在文本输入字段中输入角色名称以确认删除。然后选择 Delete(删除)。

删除 IAM 用户的客户托管策略

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Policies（策略）。
3. 在策略页面上，输入您在搜索框中先决条件中创建的客户托管策略名称（例如，**tutorial-serverless-application-repository**）以筛选策略列表。选择要删除的策略名称旁边的选项按钮。
4. 选择操作，然后选择删除。
5. 在出现的文本字段中输入此策略的名称，确认要删除此策略，然后选择删除。

删除 IAM 用户

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择用户，然后选中要删除的用户旁的复选框。
3. 在页面的顶部，选择删除。
4. 在删除###?对话框中，在文本输入字段中输入用户名以确认删除用户。选择 Delete（删除）。

后续步骤

完成本教程后，您可以进一步探索以下相关用例：

- 您可以创建多个 S3 对象 Lambda 接入点，并使用预构建的 Lambda 函数启用这些接入点，根据数据访问者的业务需要，将以不同方式配置这些函数，以编校特定类型的 PII。

  每种类型的用户都代入一个 IAM 角色，且只能访问一个 S3 对象 Lambda 接入点（通过 IAM 策略进行托管）。然后，将为不同编校使用案例配置的每个 `ComprehendPiiRedactionS3ObjectLambda` Lambda 函数附加到不同的 S3 对象 Lambda 接入点。对于每个 S3 对象 Lambda 接入点，您可以拥有一个支持的 S3 接入点，以便从存储共享数据集的 S3 存储桶读取数据。

  有关如何创建 S3 存储桶策略（仅允许用户通过 S3 接入点读取存储桶）的详细信息，请参阅 配置使用接入点的 IAM 策略。

  有关如何授予用户访问 Lambda 函数、S3 接入点和 S3 对象 Lambda 接入点的权限的更多信息，请参阅为对象 Lambda 接入点配置 IAM 策略。
- 您可以构建自己的 Lambda 函数，并将 S3 对象 Lambda 与自定义的 Lambda 函数结合使用，满足您的特定数据需求。

  例如，要探索各种数据值，您可以使用 S3 对象 Lambda 和自己的 Lambda 函数，该函数使用其他 Amazon Comprehend 特征（例如实体识别、关键短语识别、情绪分析和文档分类）来处理数据。您也可以使用 S3 对象 Lambda 与 Amazon Comprehend Medical（符合 HIPAA 条件的 NLP 服务），用于以上下文感知的方式分析和提取数据。

  有关如何使用 S3 对象 Lambda 和自己的 Lambda 函数转换数据的详细信息，请参阅 教程：使用 S3 对象 Lambda 转换应用程序的数据。

教程：使用 Amazon S3、Amazon CloudFront 和 Amazon Route 53 托管点播流视频

您可以将 Amazon S3 与 Amazon CloudFront 结合使用托管视频，实现以安全和可扩展的方式进行点播观看。点播视频 (VOD) 流代表您的视频内容存储在服务器上，观看者能够随时观看。

CloudFront 是一项快速、高度安全和可编程内容传送网络 (CDN) 服务。CloudFront 可以通过 HTTPS 从全球所有 CloudFront 节点安全地交付您的内容。有关 CloudFront 的更多信息，请参阅 Amazon CloudFront 开发人员指南中的什么是 Amazon CloudFront？。

CloudFront 缓存减少了源服务器必须直接响应的请求数量。观看者（最终用户）请求您使用 CloudFront 提供的视频时，请求将路由到离观看者所在位置更近的附近边缘站点。CloudFront 从其缓存中提供视频，只有在尚未缓存的情况下才从 S3 存储桶中检索视频。这种缓存管理特征可以通过低延迟、高吞吐量和高传输速度加快向全球观众传送视频的速度。有关 CloudFront 缓存管理的更多信息，请参阅 Amazon CloudFront 开发人员指南中的优化缓存和可用性。

目标

在本教程中，您将配置 S3 存储桶以托管点播视频流，用 CloudFront 进行传输，并用 Amazon Route 53 进行域名系统（DNS）和自定义域管理。

主题

先决条件：用 Route 53 注册并配置自定义域

开始本教程之前，您必须用 Route 53 注册和配置自定义域（例如，**example.com**），从而可将 CloudFront 分配配置为以后使用自定义域名。

如果没有自定义域名，并可通过 CloudFront 从类似于以下内容的 URL 中公开访问和托管您的 S3 视频：

```
https://CloudFront distribution domain name/Path to an S3 video
```

例如，**https://d111111abcdef8.cloudfront.net/sample.mp4**。

将您的 CloudFront 分配配置来使用配置了 Route 53 的自定义域名后，可以公开访问您的 S3 视频并通过 CloudFront 托管，URL 看起来类似于以下内容：

```
https://CloudFront distribution alternate domain name/Path to an S3 video
```

例如，**https://www.example.com/sample.mp4**。自定义域名对于观看者来说更简单、更直观。

若要注册自定义域，请参阅 Amazon Route 53 开发人员指南中的使用 Route 53 注册新域名。

用 Route 53 注册域名时，Route 53 会为您创建稍后在本教程中使用的托管区域。此托管区域用于存储有关如何为域路由流量的信息，例如路由到 Amazon EC2 实例或 CloudFront 分配。

您的域名注册、您的托管区域和域名收到的 DNS 查询会产生相关费用。有关更多信息，请参阅 Amazon Route 53 定价。

> **Note**
>
> 注册域名时，会立即产生费用，这是不可逆转的。您可以选择不自动续订域名，但要预先支付域名年费。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的注册新域。

步骤 1：创建 S3 存储桶

创建存储桶来存储计划流式传输的原始视频。

创建存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择创建存储桶。

   此时将打开创建存储桶页面。
4. 对于存储桶名称，输入您的存储桶的名称（例如 **`tutorial-bucket`**）。

   有关 Amazon S3 存储桶命名的更多信息，请参阅 存储桶命名规则。
5. 对于区域，选择要放置存储桶的 AWS 区域。

   如果可能，您应选择最接近大多数观看者的区域位置。有关存储桶区域的更多信息，请参阅 存储桶概述。
6. 对于阻止此存储桶的公有访问设置，保留默认设置（阻止全部公有访问权限已启用）。

   即使启用阻止全部公有访问，观看者仍然可以通过 CloudFront 访问上传的视频。此特征使用 CloudFront 托管存储在 S3 中视频的主要优势。

   除非需要为您的用例关闭一个或多个设置，我们建议您启用所有设置。有关阻止公有访问的更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。
7. 对于其余设置，保留默认值。

（可选）如果要为您的特定用例配置其他存储桶设置，请参阅 创建存储桶。

8. 选择创建存储桶。

步骤 2：将视频上传到 S3 存储桶

以下过程介绍了如何使用控制台将视频文件上传到 S3 存储桶。如果将许多大型视频文件上传到 S3 时，您还可以使用 Amazon S3 Transfer Acceleration 配置快速安全的文件传输。Transfer Acceleration 可以加快视频上传到 S3 存储桶的速度，以便远距离传输较大的视频。有关更多信息，请参阅 使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输。

将文件上传到存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择您在步骤 1中创建的存储桶名称（例如，**`tutorial-bucket`**）上传您的文件。
4. 在存储桶的对象 选项卡上，选择上传。
5. 在上传页面的文件和文件夹下，选择添加文件。
6. 选择要上传的文件，然后选择打开。

   例如，您可以上传名为`sample.mp4`的视频文件。
7. 选择上传。

步骤 3：创建 CloudFront 源访问身份

要限制从 S3 存储桶直接访问视频，请创建一个称为源访问身份（OAI）的特殊 CloudFront 用户。您会在本教程的后面部分中将 OAI 与您的分配关联。通过使用 OAI，您可以确保观看者无法绕过 CloudFront 并直接从 S3 存储桶获取视频。只有 CloudFront OAI 才能访问 S3 存储桶中的文件。有关更多信息，请参阅 Amazon CloudFront 开发人员指南中的使用 OAI 限制对 Amazon S3 内容的访问。

创建 CloudFront OAI

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。

2. 在左侧导航窗格的安全部分，选择来源访问。
3. 在身份选项卡下，选择创建来源访问身份。
4. 输入名称（例如，**S3-OAI**）作为新的源访问身份。
5. 选择创建。

步骤 4：创建 CloudFront 分配

要使用 CloudFront 在您的 S3 存储桶中提供和分发视频，您必须创建 CloudFront 分配。

分步

- 创建 CloudFront 分配。
- 查看存储桶策略

创建 CloudFront 分配。

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左侧导航窗格中，选择分发。
3. 选择创建分发。
4. 在源部分中，针对源域，选择以您在步骤 1 中创建的 S3 存储桶的名称开头的 S3 源域名（例如，**tutorial-bucket**）。
5. 对于来源访问，请选择遗留访问身份。
6. 在源访问身份下，选择您在步骤 3 中创建的现有源访问身份（例如，**S3-OAI**）。
7. 在存储桶策略下，选择是，更新存储桶策略。
8. 对于默认缓存行为部分，在观看者协议策略下，选择重新定向 HTTP 至 HTTPS。

   当您选择此特征时，HTTP 请求会自动重定向到 HTTPS，保护您的网站并保护观看者的数据。
9. 对于默认缓存行为 部分中的其他设置，请保留默认值。

   （可选）您还可以控制文件在 CloudFront 转发另一请求到您的源之前留在 CloudFront 缓存中的时长。减少持续时间让您可提供动态内容。增加持续时间意味着您的观看者将获得更好的性能，因为直接从边缘缓存提供文件的可能性更大。较长的持续时间还会减少源的负载。有关更多信息，请参阅 Amazon CloudFront 开发人员指南中的管理内容在边缘缓存中保留的时长（过期时间）。
10. 对于其他部分，将其余设置保留设置为默认值。

有关不同设置选项的更多信息，请参阅 Amazon CloudFront 开发人员指南中的[您创建或更新分发时指定的值]。

11. 在页面底部，选择创建分发。
12. 在 CloudFront 的常规选项卡中，在详细信息下，上次修改列中的分发值从部署中更改为上次修改分发的时间戳。该过程通常需要花费几分钟的时间。

查看存储桶策略

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择您之前用作 CloudFront 分配源的存储桶名称（例如，**tutorial-bucket**）。
4. 选择权限选项卡。
5. 在存储桶策略部分中，确认您看到类似于存储桶策略文本中以下内容的语句：

```
{
    "Version": "2008-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::tutorial-bucket/*"
        }
    ]
}
```

这是您之前选择是，更新存储桶策略时 CloudFront 分配添加到存储桶策略的语句。

此存储桶策略更新表示您已成功配置 CloudFront 分配以限制对 S3 存储桶的访问。由于存在此限制，只能通过 CloudFront 分配访问存储桶中的对象。

步骤 5：通过 CloudFront 分配访问视频

现在，CloudFront 可以为存储在 S3 存储桶中的视频提供服务。要通过 CloudFront 访问您的视频，您必须将 CloudFront 分配域名与 S3 存储桶中视频路径组合在一起。

用 CloudFront 分配域名创建 S3 视频的 URL

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左侧导航窗格中，选择分发。
3. 要获取分发域名，请执行以下操作：
   a. 在源列中，通过查找其源名称来识别正确的 CloudFront 分配，该名称以步骤 1 中创建的 S3 存储桶开头（例如，**tutorial-bucket**）。
   b. 从列表中找到分布后，扩大域名列，复制域名值来获取您的 CloudFront 分配。
4. 在新浏览器选项卡中，粘贴您之前复制的分发域名。
5. 返回到上一个浏览器选项卡，然后通过以下网址打开 S3 控制台：https://console.aws.amazon.com/s3/。
6. 在左侧导航窗格中，选择存储桶。
7. 在存储桶列表中，选择您在步骤 1 中创建的存储桶名称（例如，**tutorial-bucket**）。
8. 在对象列表中，选择您在步骤 2 中上传的视频名称（例如，`sample.mp4`）。
9. 在对象详细信息页面上的对象概述部分中，复制密钥值。此值是指向 S3 存储桶中上传视频对象的路径。
10. 返回您之前粘贴分发域名的浏览器选项卡，在分发域名之后输入正斜杠 (**/**)，然后粘贴您之前复制的视频的路径（例如，`sample.mp4`）。

    现在，您的 S3 视频可以公开访问，并通过 CloudFront 托管在类似于以下的 URL 中：

    ```
    https://CloudFront distribution domain name/Path to the S3 video
    ```

    使用适当的值替换 *CloudFront ####*和 *S3 #####*。示例 URL 为 **https://d111111abcdef8.cloudfront.net/sample.mp4**

步骤 6：配置您的 CloudFront 分配以使用自定义域名

要使用自己的域名而非 URL 中的 CloudFront 域名来访问 S3 视频，您必须在 CloudFront 分配中添加备用域名。

分步

- 请求 SSL 证书
- 将备用域名添加到您的 CloudFront 分配。
- 创建 DNS 记录，将备用域名的流量路由到您的 CloudFront 分配的域名
- 检查是否为您的分发启用了 IPv6，并根据需要创建另一个 DNS 记录

请求 SSL 证书

要允许观看者在视频流的 URL 中使用 HTTPS 和自定义域名，请使用 AWS Certificate Manager (ACM) 以请求安全套接字层 (SSL) 证书。SSL 证书建立了指向网站的加密网络连接。

1. 登录到 AWS Management Console 并通过 https://console.aws.amazon.com/acm/ 打开 ACM 控制台。
2. 如果显示介绍页面，请在设置证书中，选择开始。
3. 在请求证书页面上，选择请求公有证书，然后选择请求证书。
4. 在添加域名页面上，键入您要使用 SSL/TLS 证书保护的站点的完全限定域名 (FQDN)。使用星号 (*) 创建通配符证书保护同一域中的多个站点名称。具体来说，在本教程中，键入 ***** 和您在先决条件中配置的自定义域名。例如，输入 ***.example.com**，然后选择下一步。

   有关更多信息，请参阅 AWS Certificate Manager 用户指南中的请求 ACM 公有证书（控制台）。
5. 在选择验证方法页面上，选择 DNS 验证。然后选择下一步。

   如果您可以编辑 DNS 配置，建议使用 DNS 域验证而不是电子邮件验证。相对于电子邮件验证，DNS 验证有多种优势。有关更多信息，请参阅 AWS Certificate Manager 用户指南中的选项 1：DNS 验证。
6. （可选）在添加标签页面上，您可以选择用元数据标记证书。
7. 选择审核。
8. 在审核页面上，验证域名和验证方法下的信息都正确无误。然后，选择确认和请求。

   验证页面显示正在处理您的请求，并且正在验证证书域。等待验证的证书处于等待验证状态。

9. 在验证页面上，选择自定义域名左侧的向下箭头，然后选择在 Route 53 中创建记录以通过 DNS 验证您的域所有权。

   这将添加由 AWS Certificate Manager 提供的 CNAME 记录到您的 DNS 配置。

10. 在在 Route 53 中创建记录对话框中，选择创建。

    验证页面应在底部显示成功状态通知。

11. 选择继续查看证书列表页。

    新证书的状态将在 30 分钟内从等待验证更改为已颁发。

将备用域名添加到您的 CloudFront 分配。

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左侧导航窗格中，选择分发。
3. 为您在步骤 4 中创建的分发选择 ID。
4. 在常规选项卡，转到设置部分，然后选择编辑。
5. 在编辑设置页面上，对于备用域名（CNAME）- 可选，选择添加项目以添加要在此 CloudFront 分配提供的 S3 视频 URL 中使用的自定义域名。

   例如，在本教程中，如果您要路由子域，如 `www.example.com`，请输入带域名 (`example.com`) 的子域名 (`www`)。具体来说，输入 **`www.example.com`**。

   > **Note**
   >
   > 您添加的备用域名（CNAME）必须包含您之前附加到 CloudFront 分配的 SSL 证书。

6. 对于自定义 SSL 证书 - 可选，选择您在之前请求的 SSL 证书（例如，**`*.example.com`**）。

   > **Note**
   >
   > 如果在请求后没有立即看到 SSL 证书，则可以等待 30 分钟，然后刷新列表，直到 SSL 证书可供您选择。

7. 对于其余设置，保留默认值。选择 Save changes（保存更改）。
8. 在分发的常规选项卡中，请等待上次修改的值从部署中更改为上次修改分发的时间戳。

创建 DNS 记录，将备用域名的流量路由到您的 CloudFront 分配的域名

1. 登录 AWS Management Console，并通过以下网址打开 Route 53 控制台：https://console.aws.amazon.com/route53/。
2. 在左侧导航窗格中，选择托管区域。
3. 在托管区域页面上，选择 Route 53 在先决条件中为您创建的托管区域名称（例如，**example.com**）。
4. 选择创建记录，然后使用快速创建记录方法。
5. 对于记录名称，请保持记录名称的值与您在之前添加的 CloudFront 分配的备用域名相同。

   在本教程中，要将流量路由到子域，如 www.example.com，请输入不带域名的子域名。例如，在您的自定义域名之前仅输入 **www** 文本字段。
6. 在记录类型中，选择 A – 将流量路由到 IPv4 地址和某些 AWS 资源。
7. 对于值，选择别名切换以启用别名资源。
8. 在将流量路由至下，从下拉列表中选择别名到 CloudFront 分配。
9. 在选择分发搜索框中，选择您在步骤 4 中创建的 CloudFront 分配的域名。

   要查找您的 CloudFront 分配域名，请执行以下操作：

   a. 在新浏览器选项卡中，登录到 AWS Management Console 并通过 https://console.aws.amazon.com/cloudfront/v3/home 打开 CloudFront 控制台。
   b. 在左侧导航窗格中，选择分发。
   c. 在源列中，通过查找其源名称来识别正确的 CloudFront 分配，该名称以步骤 1 中创建的 S3 存储桶开头（例如，**tutorial-bucket**）。
   d. 从列表中找到分发后，扩大域名列，查找域名来获取您的 CloudFront 分配。
10. 在 Route 53 控制台中的创建记录页面上，对于其余设置，请保留默认值。
11. 选择创建记录。

检查是否为您的分发启用了 IPv6，并根据需要创建另一个 DNS 记录

如果为您的分发启用了 IPv6，则必须创建另一个 DNS 记录。

1. 要检查是否为分发启用了 IPv6，请执行以下操作：

   a. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。

b. 在左侧导航窗格中，选择分发。

c. 选择您在步骤 4 中创建的 CloudFront 分配的 ID。

d. 在常规选项卡上，在设置下，请检查是否启用 IPv6。

如果为您的分发启用了 IPv6，则必须创建另一个 DNS 记录。

2. 如果为分发启用了 IPv6，请执行以下操作来创建 DNS 记录：

a. 登录 AWS Management Console，并通过以下网址打开 Route 53 控制台：https://console.aws.amazon.com/route53/。

b. 在左侧导航窗格中，选择托管区域。

c. 在托管区域页面上，选择 Route 53 在先决条件中为您创建的托管区域名称（例如，**example.com**）。

d. 选择创建记录，然后使用快速创建记录方法。

e. 对于记录名称，在自定义域名之前的文本字段中，输入您在之前创建 IPv4 DNS 记录时键入的相同值。例如，在本教程中，要将流量路由到子域 `www.example.com`，请仅输入 **www**。

f. 对于记录类型，选择 AAAA-将流量路由到 IPv6 地址和 AWS 一些资源。

g. 对于值，选择别名切换以启用别名资源。

h. 在将流量路由至下，从下拉列表中选择别名到 CloudFront 分配。

i. 在选择分发搜索框中，选择您在步骤 4 中创建的 CloudFront 分配的域名。

j. 对于其余设置，保留默认值。

k. 选择创建记录。

步骤 7：用自定义域名通过 CloudFront 分配访问 S3 视频

要用自定义 URL 访问 S3 视频，您必须将备用域名与 S3 存储桶中视频路径结合起来。

创建自定义 URL 通过 CloudFront 分配访问 S3 视频

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左侧导航窗格中，选择分发。
3. 要获取您的 CloudFront 分配的备用域名，请执行以下操作：

a. 在源列中，通过在步骤 1 中创建的 S3 存储桶名称查找源名称来识别正确的 CloudFront 分配（例如，**tutorial-bucket**）。

b. 从列表中找到分发后，扩大备用域名列，复制 CloudFront 分配的备用域名值。

4. 在新的浏览器选项卡中，粘贴 CloudFront 分配的备用域名。
5. 返回到上一个浏览器选项卡，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
6. 查找步骤 5 中解释的 S3 视频的路径。
7. 返回到之前粘贴备用域名的浏览器选项卡，键入正斜杠 (**/**)，然后粘贴 S3 视频的路径（例如，`sample.mp4`)。

   现在，您的 S3 视频可以公开访问，并可通过 CloudFront 从类似于以下的自定义 URL 中托管：

   ```
   https://CloudFront distribution alternate domain name/Path to the S3 video
   ```

   使用适当的值替换 *CloudFront ######*和 *S3 #####*。示例 URL 为 **https://www.example.com/sample.mp4**。

（可选）步骤 8：查看有关您的 CloudFront 分配接收的请求的数据

查看有关您的 CloudFront 分配接收的请求数据

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左侧窗格中，在报告和分析中，从控制台中选择报告，范围从缓存统计数据、常见对象、顶级推荐人、使用和观看者。

   您可以筛选每个报告仪表板。有关更多信息，请参阅 Amazon CloudFront 开发人员指南中的控制台中的 CloudFront 报告。
3. 要筛选数据，请选择您在步骤 4 中创建的 CloudFront 分配的 ID。

步骤 9：清除

如果您仅使用 CloudFront 和 Route 53 托管 S3 串流视频作为学习练习，请删除您分派的 AWS 资源，使其不再产生费用。

> **Note**
>
> 注册域名时，会立即产生费用，这是不可逆转的。您可以选择不自动续订域名，但要预先支付域名年费。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的注册新域。

分步

- 删除 CloudFront 分配
- 删除 DNS 记录
- 删除您的自定义域公有托管区域
- 从 Route 53 中删除自定义域名
- 删除 S3 源存储桶中的原始视频
- 删除 S3 源存储桶

删除 CloudFront 分配

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在左侧导航窗格中，选择分发。
3. 在源列中，通过在步骤 1 中创建的 S3 存储桶名称查找以其开头的源名称，识别正确的 CloudFront 分配（例如，**`tutorial-bucket`**）。
4. 要删除 CloudFront 分配，必须先将其禁用。
   - 如果状态列的值为已启用，上次修改的值是上次修改分发的时间戳，请继续禁用分发，然后再删除它。
   - 如果状态的值为已启用，并且上次修改的值为部署中，请等待状态的值更改为上次修改分发的时间戳。然后继续禁用分发，再删除它。
5. 要禁用 CloudFront 分配，请执行以下操作：
   a. 在分发列表中，选中要删除分发的 ID 旁边的复选框。
   b. 要禁用分发，选择禁用，然后选择禁用确认。

      如果禁用的分发包含一个相关的备用域名，则 CloudFront 会停止接受该域名（例如，`www.example.com`）的流量，即便另一个分发有一个包含通配符 (*) 且匹配同样域的备用域名（例如，`*.example.com`）。

c. 状态的值立即变为禁用。等到上次修改的值从部署中变为上次修改分发的时间戳。

由于 CloudFront 必须将此更改传播到所有边缘站点，所以可能需要几分钟时间才能完成更新，并且删除选项可供您删除分发。

6. 要删除禁用的分发，请执行以下操作：

a. 选中要删除分发的 ID 旁边的复选框。

b. 选择删除，然后选择删除以确认。

删除 DNS 记录

如果要删除域的公有托管区域（包括 DNS 记录），请参阅 Amazon Route 53 开发人员指南中的 删除您的自定义域公有托管区域。如果您只想删除在步骤 6 中创建的 DNS 记录，执行以下操作：

1. 登录 AWS Management Console，并通过以下网址打开 Route 53 控制台：https://console.aws.amazon.com/route53/。
2. 在左侧导航窗格中，选择托管区域。
3. 在托管区域页面上，选择 Route 53 在先决条件中为您创建的托管区域名称（例如，**example.com**）。
4. 在记录列表中，选择要删除的记录旁边的复选框（在步骤 6 中创建的记录）。

> **Note**
>
> 您不能删除类型值为 NS或 SOA 的记录。

5. 选择删除记录。
6. 要确认删除，请选择删除。

对记录的更改需要一定时间才会传播到 Route 53 DNS 服务器。目前，验证更改是否已传播的唯一方式是使用 GetChange API 操作。更改通常在 60 秒内传播到所有 Route 53 服务器。

删除您的自定义域公有托管区域

> **Warning**
>
> 如果您希望保留域注册，但停止将互联网流量路由到您的网站或 Web 应用程序，建议您删除托管区域（如前一节所述）中的记录，而不是删除托管区域。

如果删除托管区域，其他人可以劫持域并使用您的域名将流量路由到他们自己的资源。
此外，如果您删除托管区域，则无法取消删除它。您必须创建新的托管区域并更新域注册的名称服务器，这最多需要 48 个小时生效。
如果您想要使域在互联网上不可用，可以将 DNS 服务转移到免费的 DNS 服务，然后删除 Route 53 托管区域。这可防止以后的 DNS 查询可能被错误路由。

1. 如果已经用 Route 53 注册域，请参阅 Amazon Route 53 开发人员指南中的为域添加或更改名称服务器和粘附记录，了解有关将 Route 53 名称服务器替换为新 DNS 服务的名称服务器的信息。
2. 如果此域已向其他注册商注册，请使用注册商提供的方法更改此域的名称服务器。

> **Note**
>
> 如果要删除子域 (`www.example.com`) 的托管区域，则不需要更改域 (`example.com`) 的名称服务器。

1. 登录 AWS Management Console，并通过以下网址打开 Route 53 控制台：https://console.aws.amazon.com/route53/。
2. 在左侧导航窗格中，选择托管区域。
3. 在托管区域页面上，选择您要删除的托管区域所对应的行。
4. 在托管区域的记录选项卡上，请确认您要删除的托管区域仅包含 NS 和 SOA 记录。

   如果它包含其他记录，请先将其删除。

   如果您为子域的托管区域创建了任何 NS 记录，请也删除此类记录。
5. 在托管区的 DNSSEC 签署选项卡上，禁用 DNSSEC 签名（如果启用）。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的禁用 DNSSEC 签名。
6. 在托管区域的详细信息页面顶部，选择删除区域。
7. 若要确认删除，请输入 **`delete`**，然后选择删除。

从 Route 53 中删除自定义域名

对于大多数顶级域 (TLD)，可在不需要域时删除域注册。如果在注册的计划到期时间之前从 Route 53 删除域名注册，AWS 将不退还注册费。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的删除域名注册。

> **Important**
>
> 如果您想在 AWS 账户 之间转移域或将域转移到另一个注册商，请不要删除域并期望立即重新注册。相反，请参阅 Amazon Route 53 开发人员指南中的适用文档：
>
> - 将域转移到其他 AWS 账户
> - 将域从 Amazon Route 53 转移到另一注册商

删除 S3 源存储桶中的原始视频

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择您在步骤 2 中将视频上传到其中的存储桶名称（例如，**tutorial-bucket**）。
4. 在对象选项卡上，选中要删除的对象名称左侧的复选框（例如，`sample.mp4`）。
5. 选择删除。
6. 在永久删除对象？下方，输入 **permanently delete** 以确认您要删除此对象。
7. 选择删除对象。

删除 S3 源存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择您在步骤 1 中创建的存储桶名称旁边的选项按钮（例如，**tutorial-bucket**）。
4. 选择删除。

5. 在删除存储桶页面上，通过在文本字段中输入存储桶名称来确认要删除存储桶，然后选择删除存储桶。

后续步骤

完成本教程后，您可以进一步探索以下相关使用案例：

- 将 S3 视频转换为特定电视或连接设备所需的流媒体格式，然后再使用 CloudFront 分配托管这些视频。

  要使用 Amazon S3 批量操作、AWS Lambda 和 AWS Elemental MediaConvert 将视频集批量转码为各种输出媒体格式，请参阅教程：使用 S3 批量操作、AWS Lambda 和 AWS Elemental MediaConvert 对视频进行批量转码。
- 使用 CloudFront 和 Route 53 托管存储在 S3 中的其他对象，如图像、音频、动态图形、样式表、HTML、JavaScript、反应应用等。

  有关示例，请参阅 教程：使用注册到 Route 53 的自定义域配置静态网站 和 使用 Amazon CloudFront 为网站提速。
- 使用 Amazon S3 Transfer Acceleration 配置快速安全的文件传输。Transfer Acceleration 可以加快视频上传到 S3 存储桶的速度，以便远距离传输较大的视频。Transfer Acceleration 通过 CloudFront 全球分布的边缘站点和 AWS 骨干网络路由流量，提高传输性能。它还利用网络协议优化。有关更多信息，请参阅 使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输。

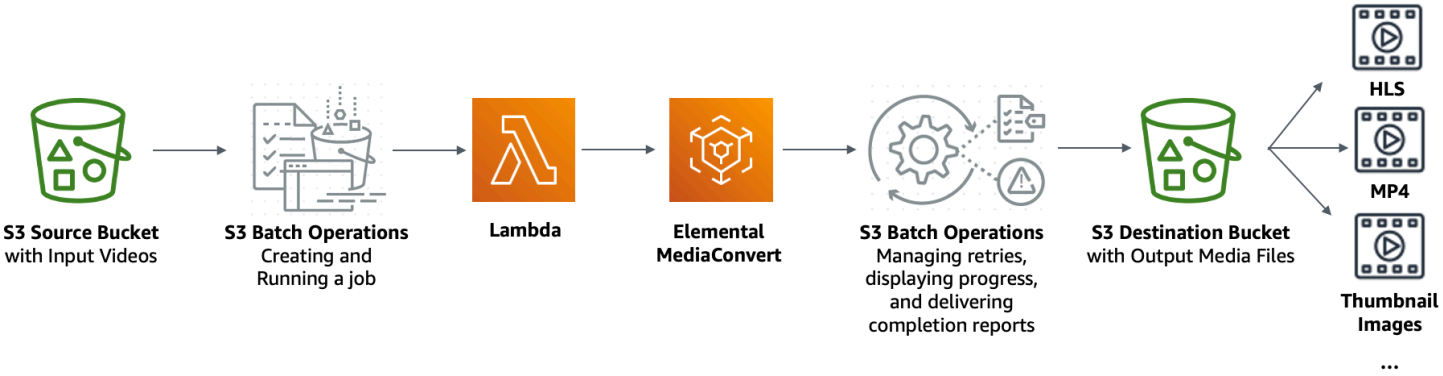
教程：使用 S3 批量操作、AWS Lambda 和 AWS Elemental MediaConvert 对视频进行批量转码

视频消费者使用各种形状、尺寸和年份的设备来享受媒体内容。这种广泛的设备类型给内容创作者和转发者带来了挑战。视频不是一刀切的格式，而是必须进行转换，以便涵盖各种尺寸、格式和比特率。如果大量必须转换的视频，这一转换任务就更具挑战性。

AWS 为您提供了构建可扩展的分布式架构的方法，该架构可执行以下操作：

- 摄取输入视频
- 处理视频以便在各种设备上播放
- 存储转码的媒体文件
- 交付输出媒体文件以满足需求

如果在 Amazon S3 中存储了大量的视频存储库，您可以将这些视频从源格式转换为多种文件类型，按照特定视频播放器或设备所需的大小、分辨率和格式转换为多种文件类型。具体来说，S3 批量操作为您提供了一个可以为 S3 源存储桶中的现有输入视频调用 AWS Lambda 函数的解决方案。然后，Lambda 函数调用 AWS Elemental MediaConvert 来执行大规模视频转码任务。转换后的输出媒体文件存储在 S3 目标存储桶中。



目标

在本教程中，您将了解如何将 S3 批量操作设置为调用 Lambda 函数对存储在 S3 源存储桶中的视频进行批量转码。Lambda 函数调用 MediaConvert 转码视频。S3 源存储桶中每个视频的输出如下所示：

- HTTP 实时流 (HLS) 自适应比特率流，用于在多个大小的设备和不同的带宽上播放。
- MP4 视频文件
- 按间隔收集的缩略图图像

主题

- 后续步骤

先决条件

在开始本教程之前，您必须有储存要转码视频的 Amazon S3 源存储桶（例如，**tutorial-bucket-1**）。

如果愿意，您可以重新命名存储桶。有关 Amazon S3 存储桶名称的更多信息，请参阅 存储桶命名规则。

对于 S3 源存储桶，请保留阻止此存储桶的公有访问设置为默认值（已启用阻止全部公共访问）。有关更多信息，请参阅 创建存储桶。

有关将视频上传到 S3 源存储桶的详细信息，请参阅 上传对象。如果将许多大型视频文件上传到 S3 时，您还可以使用 Amazon S3 Transfer Acceleration 配置快速安全的文件传输。Transfer Acceleration 可以加快视频上传到 S3 存储桶的速度，以便远距离传输较大的视频。有关更多信息，请参阅 使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输。

步骤 1：为输出媒体文件创建 S3 存储桶

在此步骤中，您将创建一个 S3 目标存储桶来存储转换后的输出媒体文件。您还可以创建跨源资源共享（CORS）配置，允许跨源访问存储在 S3 目标存储桶中的转码媒体文件。

分步

- 为输出媒体文件创建存储桶。
- 将 CORS 配置添加到 S3 输出存储桶

为输出媒体文件创建存储桶。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择创建存储桶。
4. 对于 Bucket Name（存储桶名称），输入您的存储桶的名称（例如 **tutorial-bucket-2**）。
5. 对于 Region（区域），选择要放置桶的位置 AWS 区域。
6. 要确保公众访问输出媒体文件，请在阻止此存储桶的公有访问设置，清除阻止全部公有访问权限。

> **Warning**
>
> 在完成此步骤之前，请查看 阻止对您的 Amazon S3 存储的公有访问，以确保您了解并接受允许进行公有访问所涉及的风险。当您关闭屏蔽公共访问权限设置以使您的存储桶变为公有时，Internet 上的任何人都可以访问您的存储桶。我们建议您阻止对存储桶的所有公有访问。
>
> 如果您不想清除“屏蔽公共访问权限”设置，可以使用 Amazon CloudFront 将转码的媒体文件传送给查看者（最终用户）。有关更多信息，请参阅 教程：使用 Amazon S3、Amazon CloudFront 和 Amazon Route 53 托管点播流视频。

7. 选择旁边的复选框，我确认当前设置可能会导致此存储桶及其中的对象变为公有。
8. 对于其余设置，保留默认值。
9. 选择 Create bucket（创建存储桶）。

将 CORS 配置添加到 S3 输出存储桶

JSON CORS 配置定义了在一个域中加载的客户端 Web 应用程序（此处上下文的视频播放器）在另一个域中播放转码后的输出媒体文件的方式。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Buckets（存储桶）。
3. 在 Buckets（存储桶）列表中，选择之前创建的存储桶名称（例如，**`tutorial-bucket-2`**）。
4. 选择 Permissions (权限) 选项卡。
5. 在 Cross-origin resource sharing（CORS）（跨源资源共享（CORS））部分中，请选择 Edit（编辑）。
6. 在 CORS 配置文本框中，复制并粘贴如下的 CORS 配置。

   CORS 配置必须采用 JSON 格式。在此示例中，`AllowedOrigins` 属性使用通配符 (*) 来指定所有来源。如果您知道您的特定来源，则可以限制 `AllowedOrigins` 属性添加到您的特定播放器 URL。有关配置此属性和其他属性的更多信息，请参阅 CORS 配置的元素。

```
[
    {
        "AllowedOrigins": [
```

```
            "*"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedHeaders": [
            "*"
        ],
        "ExposeHeaders": []

    }
]
```

7. 选择 Save changes (保存更改)。

步骤 2：为 MediaConvert 创建 IAM 角色

为了使用 AWS Elemental MediaConvert 转码存储在 S3 存储桶中的输入视频，您必须有一个 AWS Identity and Access Management（IAM）服务角色以授予 MediaConvert 权限，以便从 S3 源存储桶读取视频文件以及向 S3 目标存储桶写入视频文件。当您运行转码任务时，MediaConvert 控制台会使用此角色。

为 MediaConvert 创建 IAM 角色

1. 使用您选择的角色名称创建 IAM 角色（例如，**`tutorial-mediaconvert-role`**）。要创建此角色，请按照《AWS Elemental MediaConvert 用户指南》的在 IAM（控制台）中创建您的 MediaConvert 角色中的步骤操作。
2. 为 MediaConvert 创建 IAM 角色之后，请在 Role (角色)列表中选择您为 MediaConvert 创建的角色的名称（例如，**`tutorial-mediaconvert-role`**）。
3. 在 Summary (摘要) 页面上，复制 Role ARN (角色 ARN) (从 `arn:aws:iam::` 开始) 并保存 ARN 以供稍后使用。

   有关 ARN 的更多信息，请参阅 AWS 一般参考中的 Amazon Resource Name (ARN)。

步骤 3：为您的 Lambda 函数创建 IAM 角色

要使用 MediaConvert 和 S3 Batch 操作批量转码视频，您可使用 Lambda 函数来连接这两个服务来转换视频。此 Lambda 函数必须具有一个 IAM 角色，该角色可授予 Lambda 函数访问 MediaConvert 和 S3 Batch Operations 的权限。

分步

- 为 Lambda 函数创建 IAM 角色
- 为 Lambda 函数的 IAM 角色嵌入内联策略

为 Lambda 函数创建 IAM 角色

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择角色，然后选择创建角色。
3. 选择 AWS 服务角色类型，然后在 Common use cases (常见用例)中，选择 Lambda。
4. 选择 Next: Permissions (下一步: 权限)。
5. 在 Attach permissions policies (附加权限策略) 页的 Filter policies (筛选策略) 框中输入 **`AWSLambdaBasicExecutionRole`**。若要附加托管策略 AWSLambdaBasicExecutionRole 至此角色，以向 Amazon CloudWatch Logs 授予写入权限，则应选中 AWSLambdaBasicExecutionRole 旁边的复选框。
6. 选择 Next: Tags (下一步：标签)。
7. （可选）将标签添加到托管策略。
8. 选择 Next: Review (下一步：审核)。
9. 对于 Role name（角色名称），输入 **`tutorial-lambda-transcode-role`**。
10. 选择 Create role (创建角色)。

为 Lambda 函数的 IAM 角色嵌入内联策略

您必须使用内联策略，方可向 Lambda 函数执行所需的 MediaConvert 资源授予权限。

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Roles（角色）。
3. 在 Roles (角色) 列表中，选择您在前面为 Lambda 函数创建的 IAM 角色的名称（例如，**`tutorial-lambda-transcode-role`**）。
4. 选择 Permissions (权限) 选项卡。
5. 选择 Add inline policy (添加内联策略)。
6. 选择 JSON 选项卡，然后复制并粘贴以下 JSON 策略。

将在 JSON 策略中的示例 ARN 值 `Resource` 替换为您在步骤 2中为 MediaConvert 创建的 IAM 角色的角色 ARN（例如，**`tutorial-mediaconvert-role`**）。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "Logging"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/tutorial-mediaconvert-role"
            ],
            "Effect": "Allow",
            "Sid": "PassRole"
        },
        {
            "Action": [
                "mediaconvert:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "MediaConvertService"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "*"
```

```
            ],
            "Effect": "Allow",
            "Sid": "S3Service"
        }
    ]
}
```

7. 选择 Review Policy (查看策略)。
8. 对于 Name (名称)，请输入 **tutorial-lambda-policy**。
9. 选择 Create Policy (创建策略)。

   创建内联策略后，会自动嵌入您的 Lambda 函数 IAM 角色。

步骤 4：创建 Lambda 函数以进行视频转码

在本教程的此部分中，您使用适用于 Python 的 SDK 构建 Lambda 函数，以便与 S3 批量操作和 MediaConvert 集成。要开始对存储在 S3 源存储桶中的视频进行转码，您可运行 S3 Batch Operation，该任务直接为 S3 源存储桶中的每个视频调用 Lambda 函数。然后，Lambda 函数将每个视频的转码任务提交给 MediaConvert。

分步

- 编写 Lambda 函数代码并创建部署程序包
- 使用执行角色创建 Lambda 函数（控制台）
- 使用 .zip 文件归档部署 Lambda 函数并配置 Lambda 函数（控制台）

编写 Lambda 函数代码并创建部署程序包

1. 在您的本地计算机上，创建名为 `batch-transcode` 的文件夹。
2. 在 `batch-transcode` 文件夹中，创建具有 JSON 任务设置的文件。例如，您可以使用本部分中提供的设置并为文件指定名称 `job.json`。

   `job.json` 文件指定以下内容：

   - 要转码的文件
   - 您希望如何对输入的视频进行转码
   - 您希望创建何种输出媒体文件
   - 要为转码文件指定的名称

- 在何处保存转码文件
- 要应用的高级特征等。

在本教程中，我们使用如下 job.json 文件为 S3 源存储桶中的每个视频创建以下输出：

- HTTP 实时流 (HLS) 自适应比特率流，用于在多个大小的设备和不同的带宽上播放。
- MP4 视频文件
- 按间隔收集的缩略图图像

此示例 job.json 文件使用质量定义的可变比特率 (QVBR) 来优化视频质量。HTTP 实时流 (HLS) 输出符合苹果标准（从视频解除音频，6 秒的段持续时间，并通过自动 QVBR 优化视频质量）。

如果您不想使用此处提供的示例设置，可以基于您的使用案例生成 job.json 规范。要确保输出的一致性，请确保输入文件具有类似的视频和音频配置。对于具有不同视频和音频配置的任何输入文件，可创建单独的自动化（唯一的 job.json 设置）。有关更多信息，请参阅《AWS Elemental MediaConvert 用户指南》中的 JSON 格式的 AWS Elemental MediaConvert 任务设置示例。

```
{
  "OutputGroups": [
    {
      "CustomName": "HLS",
      "Name": "Apple HLS",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
              "AudioPids": [
```

```
                482,
                483,
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 640,
            "ScalingBehavior": "DEFAULT",
            "Height": 360,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 1200000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
```

```
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_360"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "TimedMetadataPid": 502,
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
```

```
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 960,
            "ScalingBehavior": "DEFAULT",
            "Height": 540,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 3500000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
```

```
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_540"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
                486,
```

```
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 5000000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
```

```
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_720"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {}
          },
          "AudioDescriptions": [
            {
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "Bitrate": 96000,
                  "CodingMode": "CODING_MODE_2_0",
                  "SampleRate": 48000
                }
              }
            }
          ],
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioTrackType": "ALTERNATE_AUDIO_AUTO_SELECT_DEFAULT"
```

```
            }
          },
          "NameModifier": "_audio"
        }
      ],
      "OutputGroupSettings": {
        "Type": "HLS_GROUP_SETTINGS",
        "HlsGroupSettings": {
          "ManifestDurationFormat": "INTEGER",
          "SegmentLength": 6,
          "TimedMetadataId3Period": 10,
          "CaptionLanguageSetting": "OMIT",
          "Destination": "s3://EXAMPLE-BUCKET/HLS/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          },
          "TimedMetadataId3Frame": "PRIV",
          "CodecSpecification": "RFC_4281",
          "OutputSelection": "MANIFESTS_AND_SEGMENTS",
          "ProgramDateTimePeriod": 600,
          "MinSegmentLength": 0,
          "DirectoryStructure": "SINGLE_DIRECTORY",
          "ProgramDateTime": "EXCLUDE",
          "SegmentControl": "SEGMENTED_FILES",
          "ManifestCompression": "NONE",
          "ClientCache": "ENABLED",
          "StreamInfResolution": "INCLUDE"
        }
      }
    },
    {
      "CustomName": "MP4",
      "Name": "File Group",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "MP4",
            "Mp4Settings": {
              "CslgAtom": "INCLUDE",
              "FreeSpaceBox": "EXCLUDE",
```

```
            "MoovPlacement": "PROGRESSIVE_DOWNLOAD"
          }
        },
        "VideoDescription": {
          "Width": 1280,
          "ScalingBehavior": "DEFAULT",
          "Height": 720,
          "TimecodeInsertion": "DISABLED",
          "AntiAlias": "ENABLED",
          "Sharpness": 100,
          "CodecSettings": {
            "Codec": "H_264",
            "H264Settings": {
              "InterlaceMode": "PROGRESSIVE",
              "ParNumerator": 1,
              "NumberReferenceFrames": 3,
              "Syntax": "DEFAULT",
              "Softness": 0,
              "GopClosedCadence": 1,
              "HrdBufferInitialFillPercentage": 90,
              "GopSize": 2,
              "Slices": 2,
              "GopBReference": "ENABLED",
              "HrdBufferSize": 10000000,
              "MaxBitrate": 5000000,
              "ParDenominator": 1,
              "EntropyEncoding": "CABAC",
              "RateControlMode": "QVBR",
              "CodecProfile": "HIGH",
              "MinIInterval": 0,
              "AdaptiveQuantization": "AUTO",
              "CodecLevel": "AUTO",
              "FieldEncoding": "PAFF",
              "SceneChangeDetect": "ENABLED",
              "QualityTuningLevel": "SINGLE_PASS_HQ",
              "UnregisteredSeiTimecode": "DISABLED",
              "GopSizeUnits": "SECONDS",
              "ParControl": "SPECIFIED",
              "NumberBFramesBetweenReferenceFrames": 3,
              "RepeatPps": "DISABLED",
              "DynamicSubGop": "ADAPTIVE"
            }
          },
          "AfdSignaling": "NONE",
```

```
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "AudioDescriptions": [
            {
              "AudioTypeControl": "FOLLOW_INPUT",
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "AudioDescriptionBroadcasterMix": "NORMAL",
                  "Bitrate": 160000,
                  "RateControlMode": "CBR",
                  "CodecProfile": "LC",
                  "CodingMode": "CODING_MODE_2_0",
                  "RawFormat": "NONE",
                  "SampleRate": 48000,
                  "Specification": "MPEG4"
                }
              },
              "LanguageCodeControl": "FOLLOW_INPUT",
              "AudioType": 0
            }
          ]
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/MP4/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    },
    {
      "CustomName": "Thumbnails",
      "Name": "File Group",
```

```
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "RAW"
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "FRAME_CAPTURE",
              "FrameCaptureSettings": {
                "FramerateNumerator": 1,
                "FramerateDenominator": 5,
                "MaxCaptures": 500,
                "Quality": 80
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          }
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/Thumbnails/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    }
  ],
  "AdAvailOffset": 0,
```

```
  "Inputs": [
    {
      "AudioSelectors": {
        "Audio Selector 1": {
          "Offset": 0,
          "DefaultSelection": "DEFAULT",
          "ProgramSelection": 1
        }
      },
      "VideoSelector": {
        "ColorSpace": "FOLLOW"
      },
      "FilterEnable": "AUTO",
      "PsiControl": "USE_PSI",
      "FilterStrength": 0,
      "DeblockFilter": "DISABLED",
      "DenoiseFilter": "DISABLED",
      "TimecodeSource": "EMBEDDED",
      "FileInput": "s3://EXAMPLE-INPUT-BUCKET/input.mp4"
    }
  ]
}
```

3. 在 batch-transcode 文件夹中，创建 Lambda 函数的文件。您可以使用以下 Python 示例并将文件命名为 convert.py。

   S3 Batch Operation 将特定任务数据发送到 Lambda 函数，并要求返回结果数据。对于 Lambda 函数的请求和响应示例、响应和结果代码的信息以及 S3 Batch Operations 的示例 Lambda 函数的信息，请参阅 调用 AWS Lambda 函数。

```
import json
import os
from urllib.parse import urlparse
import uuid
import boto3

"""
When you run an S3 Batch Operations job, your job
invokes this Lambda function. Specifically, the Lambda function is
invoked on each video object listed in the manifest that you specify
for the S3 Batch Operations job in Step 5.

Input parameter "event": The S3 Batch Operations event as a request
```

```
                          for the Lambda function.

Input parameter "context": Context about the event.

Output: A result structure that Amazon S3 uses to interpret the result
        of the operation. It is a job response returned back to S3 Batch
 Operations.
"""
def handler(event, context):

    invocation_schema_version = event['invocationSchemaVersion']
    invocation_id = event['invocationId']
    task_id = event['tasks'][0]['taskId']

    source_s3_key = event['tasks'][0]['s3Key']
    source_s3_bucket = event['tasks'][0]['s3BucketArn'].split(':::')[-1]
    source_s3 = 's3://' + source_s3_bucket + '/' + source_s3_key

    result_list = []
    result_code = 'Succeeded'
    result_string = 'The input video object was converted successfully.'

    # The type of output group determines which media players can play
    # the files transcoded by MediaConvert.
    # For more information, see Creating outputs with AWS Elemental MediaConvert.
    output_group_type_dict = {
        'HLS_GROUP_SETTINGS': 'HlsGroupSettings',
        'FILE_GROUP_SETTINGS': 'FileGroupSettings',
        'CMAF_GROUP_SETTINGS': 'CmafGroupSettings',
        'DASH_ISO_GROUP_SETTINGS': 'DashIsoGroupSettings',
        'MS_SMOOTH_GROUP_SETTINGS': 'MsSmoothGroupSettings'
    }

    try:
        job_name = 'Default'
        with open('job.json') as file:
            job_settings = json.load(file)

        job_settings['Inputs'][0]['FileInput'] = source_s3

        # The path of each output video is constructed based on the values of
        # the attributes in each object of OutputGroups in the job.json file.
        destination_s3 = 's3://{0}/{1}/{2}' \
            .format(os.environ['DestinationBucket'],
```

```
                    os.path.splitext(os.path.basename(source_s3_key))[0],
                    os.path.splitext(os.path.basename(job_name))[0])

        for output_group in job_settings['OutputGroups']:
            output_group_type = output_group['OutputGroupSettings']['Type']
            if output_group_type in output_group_type_dict.keys():
                output_group_type = output_group_type_dict[output_group_type]
                output_group['OutputGroupSettings'][output_group_type]
['Destination'] = \
                    "{0}{1}".format(destination_s3,
                                    urlparse(output_group['OutputGroupSettings']
[output_group_type]['Destination']).path)
            else:
                raise ValueError("Exception: Unknown Output Group Type {}."
                                 .format(output_group_type))

        job_metadata_dict = {
            'assetID': str(uuid.uuid4()),
            'application': os.environ['Application'],
            'input': source_s3,
            'settings': job_name
        }

        region = os.environ['AWS_DEFAULT_REGION']
        endpoints = boto3.client('mediaconvert', region_name=region) \
            .describe_endpoints()
        client = boto3.client('mediaconvert', region_name=region,
                              endpoint_url=endpoints['Endpoints'][0]['Url'],
                              verify=False)

        try:
            client.create_job(Role=os.environ['MediaConvertRole'],
                              UserMetadata=job_metadata_dict,
                              Settings=job_settings)
        # You can customize error handling based on different error codes that
        # MediaConvert can return.
        # For more information, see MediaConvert error codes.
        # When the result_code is TemporaryFailure, S3 Batch Operations retries
        # the task before the job is completed. If this is the final retry,
        # the error message is included in the final report.
        except Exception as error:
            result_code = 'TemporaryFailure'
            raise
```

```
    except Exception as error:
        if result_code != 'TemporaryFailure':
            result_code = 'PermanentFailure'
        result_string = str(error)

    finally:
        result_list.append({
            'taskId': task_id,
            'resultCode': result_code,
            'resultString': result_string,
        })

    return {
        'invocationSchemaVersion': invocation_schema_version,
        'treatMissingKeyAs': 'PermanentFailure',
        'invocationId': invocation_id,
        'results': result_list
    }
```

4. 若要在本地终端中使用 `convert.py` 和 `job.json` 创建部署程序包 (作为名为 `lambda.zip` 的 `.zip` 文件)，打开之前创建的 `batch-transcode` 文件夹，并运行以下命令。

   对于 macOS 用户，运行以下命令：

   ```
   zip -r lambda.zip convert.py job.json
   ```

   对于 Windows 用户，运行以下命令：

   ```
   powershell Compress-Archive convert.py lambda.zip
   ```

   ```
   powershell Compress-Archive -update job.json lambda.zip
   ```

使用执行角色创建 Lambda 函数（控制台）

1. 通过 https://console.aws.amazon.com/lambda/ 打开 AWS Lambda 控制台。
2. 在左侧导航窗格中，选择 Functions (函数)。
3. 选择 Create function (创建函数)。
4. 选择从头开始创作。

5. 在基本信息中，执行以下操作：

   a. 对于 Function name (函数名称)，请输入 **tutorial-lambda-convert**。

   b. 对于 Runtime (运行时)，选择 Python 3.8 或更高的 Python 版本。

6. 选择 Change default execution role (更改默认执行角色)，在 Execution role (执行角色)中，选择 Use an existing role (使用现有角色)。

7. 在 Existing role (退出角色) 中，选择步骤 3中您为 Lambda 函数创建的 IAM 角色的名称（例如，**tutorial-lambda-transcode-role**)。

8. 对于其余设置，保留默认值。

9. 选择 Create function（创建函数）。

使用 .zip 文件归档部署 Lambda 函数并配置 Lambda 函数（控制台）

1. 在您创建的 Lambda 函数代码源部分中（例如，**tutorial-lambda-convert**），请选择上传自，然后选择 .zip 文件。

2. 选择上传以选择本地 `.zip` 文件。

3. 选择您之前创建的 `lambda.zip` 文件，然后选择打开。

4. 选择保存。

5. 在运行时间设置部分，选择编辑。

6. 要告诉 Lambda 运行时要调用 Lambda 函数代码中的哪个处理程序方法，请输入 **convert.handler** 的处理程序字段。

   当您在 Python 中配置函数时，处理程序设置的值是文件名，也是处理程序模块的名称（由点 (.) 分隔）。例如，`convert.handler` 调用在 `convert.py` 文件中定义的 `handler` 方法。

7. 选择保存。

8. 在 Lambda 函数页面上，选择配置选项卡。在左侧导航窗格中的配置选项卡上，选择环境变量，然后选择编辑。

9. 选择 Add environment variable (添加环境变量)。然后，输入以下每个环境变量的指定密钥和值。

   - 密钥：**DestinationBucket** 值：**tutorial-bucket-2**

     此值是您在步骤 1 中创建的输出媒体文件的 S3 存储桶。

   - 密钥：**MediaConvertRole** 值：**arn:aws:iam::*111122223333*:role/tutorial-mediaconvert-role**

此值是您在步骤 2 中创建的 MediaConvert IAM 角色 ARN。）确保用 IAM 角色的实际 ARN 替换此 ARN。

- 密钥：**Application** 值：**Batch-Transcoding**

  此值是您的应用程序的名称。

10. 选择保存。
11. （可选）在配置选项卡上，在左侧导航窗格的常规配置部分中，选择编辑。在超时字段中，输入 **2** 分钟 **0** 秒。然后，选择保存。

    超时 –是Lambda允许函数在停止调用之前为调用运行的时长。默认值为 3 秒。定价基于配置的内存量和代码运行的时间量。有关更多信息，请参阅AWS Lambda 定价。

步骤 5：为 S3 源存储桶配置 Amazon S3 清单

设置 Lambda 函数转码后，创建 S3 Batch Operation 以转码一组视频。首先，您需要您希望 S3 批量操作对其运行指定转码操作的输入视频对象列表。要获取输入视频对象列表，您可以为 S3 源存储桶生成 S3 清单报告（例如 **tutorial-bucket-1**）。

分步

- 为输入视频的 S3 清单报告创建和配置存储桶
- 为 S3 视频源存储桶配置 Amazon S3 清单
- 检查 S3 视频源存储桶清单报告

为输入视频的 S3 清单报告创建和配置存储桶

要存储列出 S3 源存储桶对象的 S3 清单报告，请创建 S3 清单目标存储桶，然后为存储桶配置存储桶策略，将清单文件写入 S3 源存储桶。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择创建存储桶。
4. 对于 Bucket Name（存储桶名称），输入您的存储桶的名称（例如 **tutorial-bucket-3**）。
5. 对于 AWS 区域，选择要中放置存储桶的 AWS 区域。

目标存储桶必须位于与您为其设置 S3 清单存储桶相同的 AWS 区域 中。清单目标存储桶可处于不同的 AWS 账户中。

6. 在阻止此存储桶的公有访问设置中，保留默认设置（已启用阻止全部公有访问）。
7. 对于其余设置，保留默认值。
8. 选择 Create bucket（创建存储桶）。
9. 在存储桶列表中，选择刚刚创建的存储桶名称（例如，**tutorial-bucket-3**）。
10. 要授予 Amazon S3 将清单报告的数据写入 S3 清单目标存储桶的权限，您可选择权限选项卡。
11. 向下滚动到存储桶策略部分，然后选择编辑。存储桶策略页面随即打开。
12. 要授予 S3 清单的权限，请在策略字段中，粘贴以下存储桶策略。

    将三个示例值替换为以下值：

    - 您为存储库存报告创建的存储桶名称（例如，*tutorial-bucket-3*）。
    - 存储输入视频的源存储桶名称（例如，*tutorial-bucket-1*）。
    - 您用于创建 S3 视频源存储桶的 AWS 账户 ID（例如，*111122223333*）。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"InventoryAndAnalyticsExamplePolicy",
      "Effect":"Allow",
      "Principal": {"Service": "s3.amazonaws.com"},
      "Action":"s3:PutObject",
      "Resource":["arn:aws:s3:::tutorial-bucket-3/*"],
      "Condition": {
          "ArnLike": {
              "aws:SourceArn": "arn:aws:s3:::tutorial-bucket-1"
           },
         "StringEquals": {
             "aws:SourceAccount": "111122223333",
             "s3:x-amz-acl": "bucket-owner-full-control"
          }
       }
    }
  ]
}
```

13. 选择 Save changes（保存更改）。

为 S3 视频源存储桶配置 Amazon S3 清单

若要生成视频对象和元数据的平面文件列表，您必须为 S3 视频源存储桶配置 S3 清单。这些计划清单可以包括存储桶中的所有对象，或者按共享前缀分组的对象。在本教程中，S3 清单报告包含 S3 源存储桶中的所有视频对象。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 要配置 S3 源存储桶中输入视频的 S3 清单报告，请在存储桶列表中，选择 S3 源存储桶的名称（例如，**tutorial-bucket-1**）。
4. 选择管理选项卡。
5. 向下滚动到清单配置部分，然后选择创建清单配置。
6. 对于清单配置名称，输入名称（例如，**tutorial-inventory-config**）。
7. 在清单范围下，为目标版本选择仅当前版本并保留其他清单范围设置为本教程的默认值。
8. 在报告详细信息下，对于目标存储桶，选择此账户。
9. 对于目的地，选择浏览 S3，然后选择您在上面创建的目标存储桶，以将库存报告保存到其中（例如，**tutorial-bucket-3**）。然后，选择选择路径。

   目标存储桶必须位于与您为其设置 S3 清单存储桶相同的 AWS 区域 中。清单目标存储桶可处于不同的 AWS 账户中。

   在目标存储桶字段下目标存储桶权限)添加到目标存储桶策略中允许 Amazon S3 在该存储桶中放置数据。有关更多信息，请参阅 创建目标存储桶策略。
10. 在 频率下，选择每天。
11. 为输出格式选择 CSV。
12. 对于状态，选择已启用。
13. 在服务器端加密部分下，对于本教程选择禁用。

    有关更多信息，请参阅 使用 S3 控制台配置清单 和 向 Amazon S3 授予权限以使用客户自主管理型密钥进行加密。
14. 在其他字段 - 可选部分中，选择大小、上次修改和存储类。
15. 选择 Create（创建）。

有关更多信息，请参阅 使用 S3 控制台配置清单。

检查 S3 视频源存储桶清单报告

在清单列表发布后，清单文件将发送到 S3 清单目标存储桶。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择视频源存储桶的名称（例如，**tutorial-bucket-1**）。
4. 选择管理。
5. 要查看 S3 清单报告是否准备好，以便您在步骤 7 中创建 S3 Batch Operation 任务，则在清单配置下，检查是否启用从清单创建任务按钮。

   > **Note**
   >
   > 交付第一份清单报告可能需要最多 48 小时。如果从清单创建任务按钮处于禁用状态，则第一个清单报告尚未送达。等到第一个清单报告交付，并且从清单创建任务按钮在创建步骤 7中的 S3 Batch Operation 任务之前已启用。

6. 要检查 S3 清单报告（`manifest.json`），在目标列中，选择您在前面创建的用于存储库存报告的库存目标存储桶名称（例如，**tutorial-bucket-3**）。
7. 在对象选项卡上，选择具有 S3 源存储桶名称的现有文件夹（例如，**tutorial-bucket-1**）。然后选择之前创建库存配置时在清单配置名称中输入的名称（例如，**tutorial-inventory-config**）。

   您可以查看以报告生成日期作为其名称的文件夹列表。
8. 要检查某一日期的每日 S3 清单报告，请选择带相应生成日期名称的文件夹，然后选择 `manifest.json`。
9. 要查看特定日期库存报告的详细信息，请在 manifest.json 页面上，选择下载或打开。

步骤 6：为 S3 批量操作创建 IAM 角色

要使用 S3 Batch Operation 执行批处理转码，您必须首先创建 IAM 角色，让 Amazon S3 拥有执行 S3 批处理操作的权限。

分步

- 为 S3 批量操作创建 IAM 策略
- 创建 S3 Batch Operations IAM 角色并附加权限策略。

为 S3 批量操作创建 IAM 策略

您必须创建 IAM 策略，为 S3 Batch Operation 授予读取输入清单、调用 Lambda 函数以及编写 S3 Batch Operation 任务完成报告的权限。

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Policies（策略）。
3. 选择创建策略。
4. 选择 JSON 选项卡。
5. 在 JSON 文本字段中，粘贴以下 JSON 策略。

   在 JSON 策略中，将四个示例值替换为以下值：

   - 存储输入视频的源存储桶名称（例如，*tutorial-bucket-1*）。
   - 您在步骤 5 中创建以存储 `manifest.json` 文件的清单目标存储桶名称（例如，*tutorial-bucket-3*）。
   - 您在步骤 1 中创建以存储输出媒体文件的存储桶名称（例如，*tutorial-bucket-2*）。在本教程中，我们将任务完成报告存储在输出媒体文件的目标存储桶中。
   - 您在步骤 4 中创建的 Lambda 函数的角色 ARN。要查找并复制 Lambda 函数的角色 ARN，请执行以下操作：
     - 在新浏览器选项卡中，打开 https://console.aws.amazon.com/lambda/home#/functions 中的 Lambda 控制台上的函数页面。
     - 在函数列表中，选择在步骤 4 中创建的 Lambda 函数（例如，**tutorial-lambda-convert**）。
     - 选择复制 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Get",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::tutorial-bucket-1/*",
                "arn:aws:s3:::tutorial-bucket-3/*"
            ]
        },
        {
            "Sid": "S3PutJobCompletionReport",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::tutorial-bucket-2/*"
        },
        {
            "Sid": "S3BatchOperationsInvokeLambda",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-west-2:111122223333:function:tutorial-lambda-
convert"
            ]
        }
    ]
}
```

6. 选择下一步：标签。
7. 选择下一步：审核。
8. 在名称字段中，输入 **tutorial-s3batch-policy**。
9. 选择创建策略。

创建 S3 Batch Operations IAM 角色并附加权限策略。

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择角色，然后选择创建角色。

3. 选择 AWS 服务 角色类型，然后选择 S3 服务。
4. 在选择您的用例中，选择 S3 批量操作。
5. 选择下一步：权限。
6. 在 Attach permissions policies（附加权限策略）下，于搜索框中输入您以前创建的 IAM 策略的名称（例如，**tutorial-s3batch-policy**) 以筛选策略列表。选中策略名称旁边的复选框（例如，**tutorial-s3batch-policy**)。
7. 选择下一步：标签。
8. 请选择下一步：审核。
9. 对于 Role name（角色名称），输入 **tutorial-s3batch-role**。
10. 选择 Create role（创建角色）。

    创建 S3 批量操作的 IAM 角色后，以下信任策略将自动附加到该角色。此信任策略允许 S3 批量操作服务主体担任 IAM 角色。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

步骤 7：设置并运行 S3 批量操作任务

要创建 S3 批量操作任务以处理 S3 源存储桶中的输入视频，您必须为此特定任务指定参数。

> **Note**
>
> 在开始创建 S3 批量操作任务之前，请确保启用从清单创建任务按钮。有关更多信息，请参阅检查 S3 视频源存储桶清单报告。如果从清单创建任务按钮处于禁用状态，则第一个清单报告尚未送达，您必须等到启用该按钮。在步骤 5 中为 S3 源存储桶配置 Amazon S3 清单后，交付第一个清单报告可能需要最长 48 小时。

分步

- 创建 S3 批量操作任务
- 运行 S3 批量操作任务以调用 Lambda 函数
- （可选）检查您的完成报告
- （可选）在 Lambda 控制台中监控每个 Lambda 调用
- （可选）监控 MediaConvert 控制台中的每个 MediaConvert 视频转码任务

创建 S3 批量操作任务

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择批量操作。
3. 选择创建任务。
4. 对于 AWS 区域，选择要在其中创建任务的区域。

   在本教程中，要使用 S3 批量操作任务调用 Lambda 函数，您必须在清单中引用的对象所在的 S3 视频源存储桶所在的同一区域中创建任务。
5. 在清单 部分中执行以下操作：

   a. 对于 Manifest format（清单格式），选择 S3 清单报告 (manifest.json)。

   b. 针对清单对象，选择浏览 S3查找您在步骤 5 中创建用于存储清单报告的存储桶（例如，**tutorial-bucket-3**)。在清单对象页面中，浏览对象名称直到找到特定日期的 `manifest.json` 文件。此文件列出了有关要批量转码的所有视频的信息。当您找到要使用的 `manifest.json` 文件之后，选择旁边的选项按钮。然后，选择选择路径。

   c. （可选）对于清单对象版本 ID - 可选，如果您希望使用其他版本而不是最新版本，则输入清单对象的版本 ID。
6. 选择下一步。
7. 要使用 Lambda 函数来转码选定 `manifest.json` 文件中列出的所有对象，在操作类型中，选择调用 AWS Lambda 函数。
8. 在调用 Lambda 函数部分中，执行以下操作：

   a. 选择从账户中的函数选择。

   b. 对于 Lambda 函数，选择您在步骤 4 中创建的 Lambda 函数（例如，**tutorial-lambda-convert**)。

c. 对于 Lambda 函数版本，请保留默认值 $LATEST。

9. 选择下一步。配置其他选项页面随即打开。
10. 在其他选项部分下，请保留默认设置。

    有关这些选项的详细信息，请参阅 批量操作任务请求元素。
11. 在完成报告部分，对于完成报告目的地的路径，选择浏览 S3。查找您在步骤 1 中创建以存储输出媒体文件的存储桶名称（例如，**tutorial-bucket-2**）。选择该存储桶名称旁边的选项按钮。然后，选择选择路径。

    对于其余的完成报告设置，请保留默认值。有关完成报告设置的更多信息，请参阅 批量操作任务请求元素。完成报告维护任务详细信息和执行的操作的记录。
12. 在权限部分下，选择从现有 IAM 角色中进行选择。对于 IAM 角色，选择您在步骤 6 中创建的 S3 批量操作任务的 IAM 角色（例如，**tutorial-s3batch-role**）。
13. 选择下一步。
14. 在审阅页面上，检查您的设置。然后选择创建任务。

    S3 读取 S3 批量操作任务清单后，它会将任务的状态设为等待确认运行。要查看任务状态的更新，请刷新页面。直到任务状态为等待确认运行，您才可运行任务。

运行 S3 批量操作任务以调用 Lambda 函数

运行批量操作任务以调用 Lambda 函数进行视频转码。如果任务失败，您可以检查完成报告以确定原因。

运行 S3 批量操作任务

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择批量操作。
3. 从任务列表中，在第一行上选择任务的任务 ID，这是之前创建的 S3 批量操作任务。
4. 选择运行任务。
5. 再次复核您的任务参数，并确认清单中列出的对象总数与清单中对象的数量相同。然后选择运行任务。

    您的 S3 批量操作任务页面打开。

6. 任务开始运行后，在任务页面上的状态下，检查 S3 批量操作任务的进度，例如状态、已完成百分比、成功总数（费率）、失败总数（速率）、终止日期和终止原因。

   S3 批量操作任务完成后，查看任务页面上的数据，确认已按预期完成任务。

   如果在尝试超过 1,000 个操作后，超过 50% 的 S3 批量操作任务的对象操作失败，该任务将自动失败。要检查完成报告确定失败的原因，请使用下面的可选步骤。

（可选）检查您的完成报告

您可以使用完成报告来确定哪些对象失败以及失败的原因。

要检查完成报告以了解有关失败对象的详细信息

1. 在 S3 批量操作任务的页面上，向下滚动至完成报告部分，然后选择完成报告目标下的链接。

   S3 输出目标存储桶页面打开。
2. 在对象选项卡上，选择之前创建的 S3 批量操作务以任务 ID 名称结尾的文件夹。
3. 选择结果/。
4. 选中 `.csv` 文件旁边的复选框。
5. 要查看任务报告，请选择打开或者下载。

（可选）在 Lambda 控制台中监控每个 Lambda 调用

S3 批量操作任务开始运行后，任务将为每个输入视频对象调用 Lambda 函数。S3 将每次 Lambda 调用的日志写入到 CloudWatch Logs。您可以使用 Lambda 控制台的监控控制面板来监控 Lambda 函数。

1. 通过 https://console.aws.amazon.com/lambda/ 打开 AWS Lambda 控制台。
2. 在左侧导航窗格中，选择 Functions（函数）。
3. 在函数列表中，选择在步骤 4 中创建的 Lambda 函数（例如，**`tutorial-lambda-convert`**）。
4. 选择监控选项卡。
5. 在指标下，请参阅 Lambda 函数的运行时间指标。
6. 在日志下，通过 CloudWatch Logs Insights 查看每个 Lambda 调用的日志数据。

> **Note**
>
> 将 S3 批量操作与 Lambda 函数结合使用时，会在每个对象上调用 Lambda 函数。如果您的 S3 批量操作任务很大，可以同时调用多个 Lambda 函数，从而导致 Lambda 并发峰值。
>
> 对于每个区域，每个 AWS 账户 都有 Lambda 并发配额。有关更多信息，请参阅 AWS Lambda 开发人员指南中的 AWS Lambda 函数扩展。将 Lambda 函数与 S3 批量操作结合使用的最佳做法是对 Lambda 函数本身设置并发限制。设置并发限制会使您的任务不会占用大部分 Lambda 并发，并可能会限制您账户中的其他函数。有关更多信息，请参阅 AWS Lambda 开发人员指南中的管理 Lambda 预留并发。

（可选）监控 MediaConvert 控制台中的每个 MediaConvert 视频转码任务

MediaConvert 任务执行转换媒体文件的工作。当您的 S3 批量操作任务为每个视频调用 Lambda 函数时，每个 Lambda 函数调用都会为每个输入视频创建一个 MediaConvert 转码任务。

1. 登录到 AWS Management Console，然后在 https://console.aws.amazon.com/mediaconvert/ 打开 MediaConvert 控制台。
2. 如果显示 MediaConvert 介绍页面，请选择开始。
3. 从任务列表中，查看每行以监控每个输入视频的转码任务。
4. 确定要检查的任务行，然后选择任务 ID 链接打开任务详细信息页面。
5. 在任务摘要页面，在输出中，根据浏览器支持的内容，选择 HLS、MP4 或缩略图输出的链接，转到输出媒体文件的 S3 目标存储桶。
6. 在 S3 输出目标存储桶的相应文件夹（HLS、MP4 或缩略图）中，选择输出媒体文件对象的名称。

   将打开对象的详细信息页面。
7. 在对象的详细信息页面上，在对象概述中，选择对象 URL 下的链接观看转码输出媒体文件。

步骤 8：检查 S3 目标存储桶中的输出媒体文件

若要从 S3 目标存储桶检查输出媒体文件

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 在左侧导航窗格中，选择存储桶。
3. 从存储桶列表中，选择您在步骤 1 中创建的输出媒体文件的 S3 目标存储桶名称（例如，**tutorial-bucket-2**）。
4. 在对象选项卡上，每个输入视频都有一个带输入视频名称的文件夹。每个文件夹都包含输入视频的转码输出媒体文件。

   要检查输出媒体文件中的输入视频，请执行以下操作：

   a. 选择要检查的输入视频名称文件夹。
   b. 选择Default/文件夹。
   c. 选择转码格式的文件夹（本教程中的 HLS、MP4 或缩略图）。
   d. 选择输出媒体文件的名称。
   e. 要观看转码后的文件，在对象的详细信息页面上，选择对象 URL 下的链接。

      HLS 格式的输出媒体文件被拆分为短段。要播放这些视频，您需要嵌入 `.m3u8` 文件的对象 URL 在兼容的播放器中。

步骤 9：清除

如果您只使用 S3 批量操作、Lambda 和 MediaConvert 转码视频作为学习练习，请删除 AWS 您分配的资源，以免继续产生费用。

分步

- 删除 S3 源存储桶的 S3 清单配置
- 删除 Lambda 函数
- 删除 CloudWatch 日志组
- 删除 IAM 角色以及 IAM 角色的内联策略
- 删除客户管理型 IAM 策略
- 清空 S3 存储桶
- 删除 S3 存储桶

删除 S3 源存储桶的 S3 清单配置

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择您的源存储桶名称（例如，**tutorial-bucket-1**）。
4. 选择管理选项卡。
5. 在清单配置部分中，选择您在步骤 5 中创建的清单配置旁边的选项按钮（例如，**tutorial-inventory-config**）。
6. 选择删除，然后选择确认。

删除 Lambda 函数

1. 通过 https://console.aws.amazon.com/lambda/ 打开 AWS Lambda 控制台。
2. 在左侧导航窗格中，选择 Functions（函数）。
3. 选择在步骤 4 中创建的函数旁边的复选框（例如，**tutorial-lambda-convert**）。
4. 选择操作，然后选择删除。
5. 在删除函数对话框中，选择删除。

删除 CloudWatch 日志组

1. 访问 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择日志，然后选择日志组。
3. 选中日志组旁边的复选框，该复选框的名称以在步骤 4 中创建的 Lambda 函数结尾（例如，**tutorial-lambda-convert**）。
4. 选择操作，然后选择删除日志组。
5. 在删除日志组对话框中，选择删除。

删除 IAM 角色以及 IAM 角色的内联策略

删除您在步骤 2、步骤 3 和步骤 6 中创建的 IAM 角色，执行以下操作：

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择角色，然后选中要删除的角色名称旁的复选框。
3. 在页面的顶部，选择删除。
4. 在确认对话框中，根据提示在文本输入字段中输入所需的响应，然后选择删除。

删除客户管理型 IAM 策略

若要删除您在步骤 6 中创建的客户托管 IAM 策略，执行以下操作：

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Policies（策略）。
3. 选择您在步骤 6 中创建的策略旁边的选项按钮（例如，**tutorial-s3batch-policy**）。您可以使用搜索框筛选策略列表。
4. 选择 Actions，然后选择 Delete。
5. 在文本字段中输入名称，确认您要删除的策略，然后选择删除。

清空 S3 存储桶

若要清空您在先决条件、步骤 1 和步骤 5 中创建的 S3 存储桶，执行以下操作：

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择要清空的存储桶名称旁边的选项图标，然后选择清空。
4. 在清空存储桶页面上，通过在文本字段中输入 **permanently delete** 来确认要清空存储桶，然后选择清空。

删除 S3 存储桶

若要删除您在先决条件、步骤 1 和步骤 5 中创建的 S3 存储桶，执行以下操作：

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择要删除的存储桶名称旁边的选项按钮。
4. 选择删除。
5. 在删除存储桶页面上，通过在文本字段中输入存储桶名称来确认要删除存储桶，然后选择删除存储桶。

后续步骤

完成本教程后，您可以进一步探索其他相关使用案例：

- 您可以使用 Amazon CloudFront 将转码后的媒体文件流式传输到全球各地的查看者。有关更多信息，请参阅 教程：使用 Amazon S3、Amazon CloudFront 和 Amazon Route 53 托管点播流视频。
- 将视频上传到 S3 源存储桶时，您可以对视频进行转码。为此，您可以配置 Amazon S3 事件触发器，该触发器自动调用 Lambda 函数以使用 MediaConvert 转码 S3 中的新对象。有关更多信息，请参阅 AWS Lambda 开发人员指南中的教程：使用 Amazon S3 触发器调用 Lambda函数。

教程：通过分段上传来上传对象并验证其数据完整性

分段上传允许将单个对象作为一组分段上传。每个分段都是对象数据的连续部分。您可以独立上传以及按任意顺序上传这些对象分段。如果任意分段传输失败，可以重新传输该分段且不会影响其他分段。上传完所有的对象分段后，Amazon S3 将汇集这些分段并创建对象。一般而言，如果您的对象大小达到了 100 MB，您应该考虑使用分段上传，而不是在单个操作中上传对象。有关分段上传的更多信息，请参阅使用分段上传来上传和复制对象。有关分段上传的相关限制，请参阅 Amazon S3 分段上传限制。

可以使用校验和来验证资产在复制时是否未被更改。执行校验和包括使用算法按顺序迭代文件中的每个字节。Amazon S3 提供了多个校验和选项，用于检查数据的完整性。我们建议您将这些完整性检查作为持久性最佳实践加以执行，并确认每个字节在传输过程中都未经更改。Amazon S3 还支持以下算法：SHA-1、SHA-256、CRC32 和 CRC32C。Amazon S3 使用其中一个或多个算法来计算额外的校验和值，并将其存储为对象元数据的一部分。有关校验和的更多信息，请参阅检查对象完整性。

目标

在本教程中，您将学习如何在 AWS 命令行界面（AWS CLI）中使用分段上传和额外的 SHA-256 校验和将对象上传到 Amazon S3。您还将学习如何通过计算已上传对象的 MD5 哈希值和 SHA-256 校验和，来检查对象的数据完整性。

主题

先决条件

- 在开始本教程之前，请确保您有权访问要上传到的 Amazon S3 存储桶。有关更多信息，请参阅 创建存储桶。
- 您必须安装并配置了 AWS CLI。如果未安装 AWS CLI，请参阅《AWS Command Line Interface用户指南》中的 Install or update to the latest version of the AWS CLI。
- 或者，您可以从控制台中使用 AWS CloudShell 运行 AWS CLI 命令。AWS CloudShell 是一个基于浏览器、预先经过身份验证的 Shell，您可以直接从 AWS Management Console中启动它。有关更多信息，请参阅《AWS CloudShell 用户指南》中的 What is CloudShell? 和 Getting started with AWS CloudShell。

步骤 1：创建大文件

如果您已经准备好要上传的文件，则可以在本教程中使用该文件。否则，请使用以下步骤创建一个 15 MB 的文件。有关分段上传的相关限制，请参阅 Amazon S3 分段上传限制。

创建大文件

根据您使用的操作系统，使用以下命令之一来创建文件。

Linux 或 macOS

要创建 15 MB 的文件，请打开本地终端并运行以下命令：

```
dd if=/dev/urandom of=census-data.bin bs=1M count=15
```

此命令创建一个名为 `census-data.bin` 的文件，其中填充了随机字节，大小为 15 MB。

Windows

要创建 15 MB 的文件，请打开本地终端并运行以下命令：

```
fsutil file createnew census-data.bin 15728640
```

此命令创建一个名为 census-data.bin 的文件，其中包含任意数据，大小为 15 MB（15728640 字节）。

步骤 2：将文件拆分为多个文件

要执行分段上传，必须将大文件拆分为若干较小的分段。然后，可以使用分段上传过程来上传较小的分段。此步骤演示如何将步骤 1 中创建的大文件拆分为较小的分段。以下示例使用名为 census-data.bin 的 15 MB 文件。

将大文件拆分为多个分段

Linux 或 macOS

要将大文件分成 5 MB 的分段，请使用 split 命令。打开终端并运行以下命令：

```
split -b 5M -d census-data.bin census-part
```

此命令将 census-data.bin 拆分成名为 census-part** 的 5 MB 分段，其中 ** 是从 00 开始的数字后缀。

Windows

要拆分大文件，请使用 PowerShell。打开 Powershell 并运行以下命令：

```
$inputFile = "census-data.bin"
$outputFilePrefix = "census-part"
$chunkSize = 5MB

$fs = [System.IO.File]::OpenRead($inputFile)
$buffer = New-Object byte[] $chunkSize
$fileNumber = 0

while ($fs.Position -lt $fs.Length) {
$bytesRead = $fs.Read($buffer, 0, $chunkSize)
$outputFile = "{0}{1:D2}" -f $outputFilePrefix, $fileNumber
$fileStream = [System.IO.File]::Create($outputFile)
$fileStream.Write($buffer, 0, $bytesRead)
$fileStream.Close()
$fileNumber++
```

```
}

$fs.Close()
```

此 PowerShell 脚本以 5 MB 大小的分块读取大文件，并将每个分块写入一个带有数字后缀的新文件。

运行相应的命令后，您应该会在执行该命令的目录中看到各个分段。每个分段都有一个与其分段编号对应的后缀，例如：

```
census-part00 census-part01 census-part02
```

步骤 3：创建分段上传并指定额外的校验和

要开始分段上传过程，您需要创建分段上传请求。此步骤包括启动分段上传以及为数据完整性指定额外的校验和。以下示例使用 SHA-256 校验和。如果您想要提供描述正上传的对象的任何元数据，必须在请求中提供它以便开始分段上传。

> **Note**
>
> 在本步骤和后续步骤中，本教程使用 SHA-256 额外算法。您可以选择为这些步骤使用其他的额外校验和，例如 CRC32、CRC32C 或 SHA-1。如果使用不同的算法，则必须在整个教程步骤中都使用该算法。

开始分段上传

在终端中，使用以下 `create-multipart-upload` 命令开始存储桶的分段上传。将 *amzn-s3-demo-bucket1* 替换为实际存储桶名称。另外，请用您选择的文件名替换 `census_data_file`。上传完成后，此文件名将成为对象键。

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket1 --key
 'census_data_file' --checksum-algorithm sha256
```

如果您的请求执行成功，您将看到如下 JSON 输出：

```
{
    "ServerSideEncryption": "AES256",
    "ChecksumAlgorithm": "SHA256",
    "Bucket": "amzn-s3-demo-bucket1",
```

```
    "Key": "census_data_file",
    "UploadId":
 "cNV6KCSNANFZapz1LUGPC5XwUVi1n6yUoIeSP138sNOKPeMhpKQRrbT9k0ePmgoOTCj9K83T4e2Gb5hQvNoNpCKqyb8m3
}
```

> **Note**
>
> 当您发送请求以开始分段上传时，Amazon S3 将返回具有上传 ID 的响应，此 ID 是分段上传的唯一标识符。无论您何时上传分段、列出分段、完成上传或停止上传，您都必须包括此上传 ID。后续步骤需要用到 UploadId、Key 和 Bucket 值，因此请务必保存这些值。
> 此外，如果您将分段上传与额外的校验和结合使用，则分段编号必须是连续的。如果您使用不连续的分段编号，则 complete-multipart-upload 请求可能会导致 HTTP 500 Internal Server Error。

步骤 4：上传分段上传的分段

在此步骤中，您将分段上传的各个分段上传到 S3 存储桶。使用 upload-part 命令单独上传每个分段。此过程要求为每个分段指定上传 ID、分段编号和要上传的文件。

上传分段

1. 上传分段时，除了上传 ID，还必须使用 --part-number 参数来指定分段编号。您可以选择 1 和 10000 之间的任意分段编号。分段编号在您正在上传的对象中唯一地识别分段及其位置。您选择的分段编号必须为连续序列（例如，它可以是 1、2 或 3）。如果您使用之前上传的分段的同一分段编号上传新分段，则之前上传的分段将被覆盖。
2. 使用 upload-part 命令来上传分段上传的每个分段。--upload-id 与步骤 3 中 create-multipart-upload 命令创建的输出内容相同。要上传数据的第一个分段，请使用以下命令：

   ```
   aws s3api upload-part --bucket amzn-s3-demo-bucket1 --key
    'census_data_file' --part-number 1 --body census-part00 --upload-id
    "cNV6KCSNANFZapz1LUGPC5XwUVi1n6yUoIeSP138sNOKPeMhpKQRrbT9k0ePmgoOTCj9K83T4e2Gb5hQvNoNpCKqy
    --checksum-algorithm SHA256
   ```

   完成每个 upload-part 命令后，您应该会看到类似以下示例的输出：

   ```
   {
   ```

```
    "ServerSideEncryption": "AES256",
    "ETag": "\"e611693805e812ef37f96c9937605e69\"",
    "ChecksumSHA256": "QLl8R4i4+SaJlrl8ZIcutc5TbZtwt2NwB8lTXkd3GH0="
}
```

3. 对于后续分段，相应地递增分段编号：

```
aws s3api upload-part --bucket amzn-s3-demo-bucket1 --key 'census_data_file' --
part-number <part-number> --body <file-path> --upload-id "<your-upload-id>" --
checksum-algorithm SHA256
```

例如，使用以下命令来上传第二个分段：

```
aws s3api upload-part --bucket amzn-s3-demo-bucket1 --key
 'census_data_file' --part-number 2 --body census-part01 --upload-id
 "cNV6KCSNANFZapz1LUGPC5XwUVi1n6yUoIeSP138sNOKPeMhpKQRrbT9k0ePmgoOTCj9K83T4e2Gb5hQvNoNpCKqy
 --checksum-algorithm SHA256
```

Amazon S3 将在响应的标头中返回每个已上传分段的实体标签（ETag）和额外的校验和。

4. 继续使用 `upload-part` 命令，直到上传了对象的所有分段。

步骤 5：列出分段上传的所有分段

要完成分段上传，您需要列出在这个特定分段上传任务中已经上传的所有分段。`list-parts` 命令的输出提供了诸如存储桶名称、键、上传 ID、分段编号、eTag、额外的校验和等信息。将此输出保存在文件中会很有帮助，这样您就可以在完成分段上传过程的下一步中使用它。可以使用以下方法创建名为 `parts.json` 的 JSON 输出文件。

创建列出所有分段的文件

1. 要生成包含所有已上传分段详细信息的 JSON 文件，请使用以下 `list-parts` 命令。将 ***amzn-s3-demo-bucket1*** 替换为实际的存储桶名称，并将 **<your-upload-id>** 替换为您在步骤 3 中收到的上传 ID。有关 `list-parts` 命令的更多信息，请参阅《AWS Command Line Interface 用户指南》中的 list-parts。

```
aws s3api list-parts --bucket amzn-s3-demo-bucket1 --key 'census_data_file' --
upload-id <your-upload-id> --query '{Parts: Parts[*].{PartNumber: PartNumber, ETag:
 ETag, ChecksumSHA256: ChecksumSHA256}}' --output json > parts.json
```

这样会生成一个名为 parts.json 的新文件。该文件采用 JSON 格式，包含所有已上传分段的信息。parts.json 文件包含分段上传的每个分段的基本信息，例如分段编号及其相应的 ETag 值，这些是完成分段上传过程所必需的。

2. 使用任何文本编辑器或通过终端打开 parts.json。以下是示例输出：

```
{
    "Parts": [
        {
            "PartNumber": 1,
            "ETag": "\"3c3097f89e2a2fece47ac54b243c9d97\"",
            "ChecksumSHA256": "fTPVHfyNHdv5VkR4S3EewdyioXECv7JBxN+d4FXYYTw="
        },
        {
            "PartNumber": 2,
            "ETag": "\"03c71cc160261b20ab74f6d2c476b450\"",
            "ChecksumSHA256": "VDWTa8enjOvULBAO3W2a6C+5/7ZnNjrnLApa1QVc3FE="
        },
        {
            "PartNumber": 3,
            "ETag": "\"81ae0937404429a97967dffa7eb4affb\"",
            "ChecksumSHA256": "cVVkXehUlzcwrBrXgPIM+EKQXPUvWist8mlUTCs4bg8="
        }
    ]
}
```

步骤 6：完成分段上传

在分段上传的所有分段都上传完毕并且列出分段信息后，最后一步是完成分段上传。此步骤会在 S3 存储桶中将所有上传的分段合并成单个对象。

> **Note**
>
> 您可以在调用 complete-multipart-upload 之前，通过在请求中包含 --checksum-sha256 来计算对象的校验和。如果校验和不匹配，Amazon S3 就会让请求失败。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的 complete-multipart-upload。

完成分段上传

要最终完成分段上传，请使用 complete-multipart-upload 命令。此命令需要用到步骤 5 中创建的 parts.json 文件、存储桶名称以及上传 ID。将 **<*amzn-s3-demo-bucket1*>** 替换为存储桶名称，并将 **<your-upload-id>** 替换为 parts.json 的上传 ID。

```
aws s3api complete-multipart-upload --multipart-upload file://parts.json --bucket amzn-s3-demo-bucket1 --key 'census_data_file' --upload-id <your-upload-id>
```

以下是示例输出：

```
{
    "ServerSideEncryption": "AES256",
    "Location": "https://amzn-s3-demo-bucket1.s3.us-east-2.amazonaws.com/
census_data_file",
    "Bucket": "amzn-s3-demo-bucket1",
    "Key": "census_data_file",
    "ETag": "\"f453c6dccca969c457efdf9b1361e291-3\"",
    "ChecksumSHA256": "aI8EoktCdotjU8Bq46DrPCxQCGuGcPIhJ51noWs6hvk=-3"
}
```

> **Note**
>
> 此时请勿删除各个分段文件。您还需要使用各个分段来对它们执行校验和，从而验证合并在一起的对象的完整性。

步骤 7：确认对象上传到存储桶

完成分段上传后，您可以验证对象已成功地上传到 S3 存储桶。要列出存储桶中的对象并确认新上传的文件已经存在，请使用 list-objects-v2 命令。

列出上传的对象

要列出存储桶中的对象，请使用 list-objects-v2 命令。将 ***amzn-s3-demo-bucket1*** 替换为实际存储桶名称：

```
aws s3api list-objects-v2 --bucket amzn-s3-demo-bucket1
```

此命令将返回存储桶中对象的列表。在对象列表中查找您上传的文件（例如 census_data_file）。

有关更多信息，请参阅《AWS Command Line Interface用户指南》中 list-objects-v2 命令的 Examples 部分。

步骤 8：使用 MD5 校验和验证对象完整性

在上传对象时，可以指定校验和算法以供 Amazon S3 使用。默认情况下，Amazon S3 将 MD5 字节摘要存储为对象的 ETag。对于分段上传，ETag 不是完整对象的校验和，而是每个单独分段的校验和组合。

使用 MD5 校验和验证对象完整性

1. 要检索已上传对象的 ETag，请执行 head-object 请求：

```
aws s3api head-object --bucket amzn-s3-demo-bucket1 --key census_data_file
```

以下是示例输出：

```
{
    "AcceptRanges": "bytes",
    "LastModified": "2024-07-26T19:04:13+00:00",
    "ContentLength": 16106127360,
    "ETag": "\"f453c6dccca969c457efdf9b1361e291-3\"",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {}
}
```

此 etag 的末尾附有“-3”。这表示对象是使用分段上传分为三个分段进行上传的。

2. 接下来，使用 md5sum 命令计算每个分段的 MD5 校验和。请确保提供指向分段文件的正确路径：

```
md5sum census-part*
```

以下是示例输出：

```
e611693805e812ef37f96c9937605e69 census-part00
```

```
63d2d5da159178785bfd6b6a5c635854 census-part01
95b87c7db852451bb38b3b44a4e6d310 census-part02
```

3. 对于此步骤，手动将 MD5 哈希值合并为一个字符串。然后，运行以下命令，将字符串转换为二进制并计算二进制值的 MD5 校验和：

```
echo
 "e611693805e812ef37f96c9937605e6963d2d5da159178785bfd6b6a5c63585495b87c7db852451bb38b3b44a
 | xxd -r -p | md5sum
```

以下是示例输出：

```
f453c6dccca969c457efdf9b1361e291 -
```

此哈希值应与步骤 1 中原始 ETag 值的哈希值相匹配，这可验证 census_data_file 对象的完整性。

当您指示 Amazon S3 使用其他校验和时，Amazon S3 会计算每个分段的校验和值并存储这些值。如果您想检索仍在进行的分段上传的各个分段的校验和值，可以使用 list-parts。

有关校验和如何处理分段对象的更多信息，请参阅检查对象完整性。

步骤 9：使用额外的校验和验证对象完整性

在本步骤中，教程使用 SHA-256 作为额外的校验和来验证对象完整性。如果您使用了不同的额外校验和，请改用该校验和值。

使用 SHA256 验证对象完整性

1. 在终端中运行以下命令（包括 --checksum-mode enabled 参数），以显示对象的 ChecksumSHA256 值：

```
aws s3api head-object --bucket amzn-s3-demo-bucket1 --key census_data_file --
checksum-mode enabled
```

以下是示例输出：

```
{
    "AcceptRanges": "bytes",
    "LastModified": "2024-07-26T19:04:13+00:00",
    "ContentLength": 16106127360,
    "ChecksumSHA256": "aI8EoktCdotjU8Bq46DrPCxQCGuGcPIhJ51noWs6hvk=-3",
    "ETag": "\"f453c6dccca969c457efdf9b1361e291-3\"",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {}
}
```

2. 使用以下命令将各个分段的 ChecksumSHA256 值解码为 base64，然后将其保存到名为 outfile 的二进制文件中。这些值可以在 parts.json 文件中找到。将示例 base64 字符串替换为实际的 ChecksumSHA256 值。

```
echo "QLl8R4i4+SaJlrl8ZIcutc5TbZtwt2NwB8lTXkd3GH0=" | base64 --decode >> outfile
echo "xCdgs1K5Bm4jWETYw/CmGYr+m6O2DcGfpckx5NVokvE=" | base64 --decode >> outfile
echo "f5wsfsa5bB+yXuwzqG1Bst91uYneqGD3CCidpb54mAo=" | base64 --decode >> outfile
```

3. 运行以下命令计算 outfile 的 SHA256 校验和：

```
sha256sum outfile
```

以下是示例输出：

```
688f04a24b42768b6353c06ae3a0eb3c2c50086b8670f221279d67a16b3a86f9 outfile
```

在下一步中，获取哈希值并将其转换为二进制值。此二进制值应与步骤 1 中的 ChecksumSHA256 值相匹配。

4. 将步骤 3 中的 SHA256 校验和转换为二进制，然后将其编码为 base64，来验证它与步骤 1 中的 ChecksumSHA256 值相匹配：

```
echo "688f04a24b42768b6353c06ae3a0eb3c2c50086b8670f221279d67a16b3a86f9" | xxd -r -p
 | base64
```

以下是示例输出：

```
aI8EoktCdotjU8Bq46DrPCxQCGuGcPIhJ51noWs6hvk=
```

此输出应确认 base64 输出与 `head-object` 命令输出中的 `ChecksumSHA256` 值相匹配。如果输出与校验和值匹配，则该对象有效。

Important

- 当您指示 Amazon S3 使用额外的校验和时，Amazon S3 会计算每个分段的校验和值并存储这些值。
- 如果您想检索仍在进行的分段上传的各个分段的校验和值，可以使用 `list-parts` 命令。

步骤 10：清除资源

如果要清理在本教程中创建的文件，请使用以下方法。有关删除上传到 S3 存储桶的文件的说明，请参阅删除 Amazon S3 对象。

删除步骤 1 中创建的本地文件：

要删除您为分段上传创建的文件，请从工作目录中运行以下命令：

```
rm census-data.bin census-part* outfile parts.json
```

教程：在 Amazon S3 上配置静态网站

Important

Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

您可以配置 Amazon S3 存储桶，使其功能与网站相似。本示例向您演练了在 Amazon S3 上托管网站的步骤。

> **Important**
>
> 以下教程需要禁用“屏蔽公共访问权限”。我们建议将“屏蔽公共访问权限”保持为启用状态。如果要将所有四个“屏蔽公共访问权限”设置保持为启用状态并托管静态网站，则可以使用 Amazon CloudFront 来源访问控制（OAC）。Amazon CloudFront 提供了设置安全静态网站所需的功能。Amazon S3 静态网站仅支持 HTTP 端点。Amazon CloudFront 使用 Amazon S3 的持久存储，同时提供额外的安全标头，如 HTTPS。HTTPS 通过加密正常 HTTP 请求并防范常见的网络攻击来增强安全性。有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的安全静态网站入门。

主题

步骤 1：创建存储桶

以下说明概述了如何创建存储桶以用于网站托管。有关创建存储桶的详细分步说明，请参阅创建存储桶。

创建存储桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 选择 Create bucket（创建存储桶）。
3. 输入 Bucket name（存储桶名称）（例如 **`example.com`**）。

4. 请选择要在其中创建存储桶的区域。

   请选择一个在地理上靠近您的区域可最大程度地减少延迟和成本，或满足法规要求。您选择的区域决定了您的 Amazon S3 网站端点。有关更多信息，请参阅 网站端点。

5. 要接受默认设置并创建存储桶，请选择 Create（创建）。

步骤 2：启用静态网站托管

创建存储桶后，您可以为存储桶启用静态网站托管。您可以创建新存储桶或使用现有存储桶。

启用静态网站托管

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择要为其启用静态网站托管的存储桶的名称。
3. 选择属性。
4. 在静态网站托管下，请选择编辑。
5. 请选择使用此存储桶托管网站。
6. 在静态网站托管下，请选择启用。
7. 在 Index document (索引文档) 中，输入索引文档的文件名，通常为 `index.html`。

   索引文档名称区分大小写，并且必须与您计划上传到 S3 存储桶的 HTML 索引文档的文件名完全匹配。当您为网站托管配置存储桶时，您必须指定索引文档。当对根域或任何子文件夹发出请求时，Amazon S3 将返回此索引文档。有关更多信息，请参阅 配置索引文档。

8. 要为 4XX 类错误提供您自己的自定义错误文档，请在错误文档中输入自定义错误文档文件名。

   错误文档名称区分大小写，并且必须与您计划上传到 S3 存储桶的 HTML 错误文档的文件名完全匹配。如果未指定自定义错误文档并发生错误，Amazon S3 返回默认 HTML 错误文档。有关更多信息，请参阅 配置自定义错误文档。

9. （可选）如果要指定高级重定向规则，请在 Redirection rules（重定向规则）中，输入 JSON 来描述规则。

   例如，您可以根据请求中的特定对象键名或前缀按条件路由请求。有关更多信息，请参阅 配置重新导向规则以使用高级条件重新导向。

10. 选择 Save changes（保存更改）。

Amazon S3 为您的存储桶启用静态网站托管。在页面底部的静态网站托管下，您可以看到存储桶的网站端点。

11. 在静态网站托管下，记下端点。

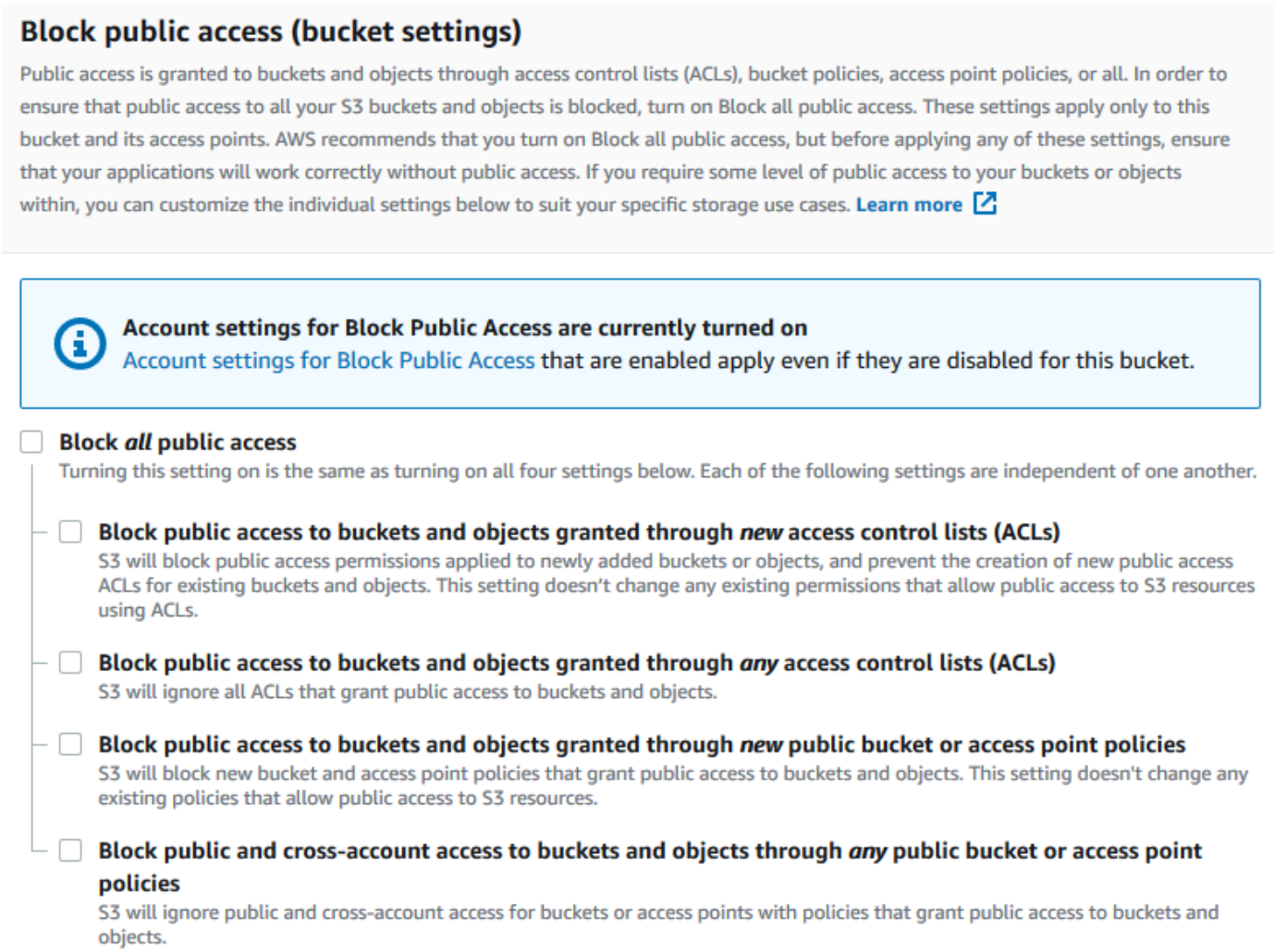
    端点是存储桶的 Amazon S3 网站端点。将存储桶配置为静态网站后，您可以使用此端点来测试您的网站。

步骤 3：编辑屏蔽公共访问权限设置

默认情况下，Amazon S3 阻止对您的账户和存储桶的公有访问权限。如果要使用存储桶托管静态网站，您可以使用以下步骤编辑您的屏蔽公共访问权限设置。

> **Warning**
>
> 在完成这些步骤之前，请查看阻止对您的 Amazon S3 存储的公有访问，来确保您了解并接受支持公共访问权限所涉及的风险。当您关闭屏蔽公共访问权限设置以使您的存储桶变为公有时，Internet 上的任何人都可以访问您的存储桶。我们建议您阻止对存储桶的所有公有访问。

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 请选择已配置为静态网站的存储桶的名称。
3. 选择权限。
4. 在屏蔽公共访问权限（存储桶设置）下，请选择编辑。
5. 清除阻止所有公有访问，然后选择保存更改。



Amazon S3 关闭了存储桶的屏蔽公共访问权限设置。要创建公有静态网站，可能还必须为您的账户配置屏蔽公共访问权限设置，然后再添加存储桶策略。如果当前已开启账户的屏蔽公共访问权限设置，您将在屏蔽公共访问权限（存储桶设置）下看到一条备注。

步骤 4：添加可使您的存储桶内容公开可用的存储桶策略

在编辑 S3 屏蔽公共访问权限设置后，您可以添加存储桶策略以授予对存储桶的公有读取访问权限。当您授予公有读取访问权限时，Internet 上的任何人都可以访问您的存储桶。

> **Important**
>
> 下面的策略仅供举例说明，仍允许完全访问您存储桶的内容。在继续执行此步骤之前，请查看如何保护 Amazon S3 存储桶中的文件？，以确保您了解保护 S3 存储桶中文件的最佳实践以及授予公有访问权限所涉及的风险。

1. 在存储桶 下，请选择存储桶的名称。
2. 选择权限。
3. 在存储桶策略下，请选择编辑。
4. 要授予对网站的公有读取访问权限，请复制以下存储桶策略，将其粘贴到存储桶策略编辑器中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. 将 Resource 更新为您的存储桶名称。

   在上述示例存储桶策略中，*Bucket-Name* 是存储桶名称的占位符。要将此存储桶策略用于您自己的存储桶，您必须更新此名称以匹配您的存储桶名称。

6. 选择保存更改。

   此时将显示一条消息，指示存储桶策略已成功添加。

   如果您看到显示 Policy has invalid resource 的错误，请确认存储桶策略中的存储桶名称与您的存储桶名称匹配。有关添加存储桶策略的信息，请参阅如何添加 S3 存储桶策略？

如果您收到错误消息且无法保存存储桶策略，请检查您的账户和存储桶的屏蔽公共访问权限设置以确认您允许对存储桶进行公有访问。

步骤 5：配置索引文档

当您为存储桶启用静态网站托管时，您可以输入索引文档的名称（例如，**index.html**）。为存储桶启用静态网站托管后，您可以将具有此索引文档名称的 HTML 文件上传到存储桶。

配置索引文档

1. 创建 index.html 文件。

   如果您没有 index.html 文件，则可以使用以下 HTML 创建一个：

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. 将索引文件保存在本地。

   索引文档文件名必须与您在静态网站托管对话框中输入的索引文档名称完全匹配。索引文档名称区分大小写。例如，如果在静态网站托管对话框中为索引文档名称输入 index.html，则索引文档文件名也必须是 index.html，而不是 Index.html。

3. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
4. 在存储桶列表中，请选择要用于托管静态网站的存储桶的名称。
5. 为您的存储桶启用静态网站托管，并输入索引文档的确切名称（例如 index.html）。有关更多信息，请参阅 启用网站托管。

   启用静态网站托管后，继续执行步骤 6。

6. 要将索引文档上传到存储桶，请执行以下操作之一：

   - 将索引文件拖放到控制台存储桶列表中。

- 选择上传，然后按照提示选择并上传索引文件。

  如需分步指导，请参阅 上传对象。

7. （可选）将其他网站内容上传到您的存储桶。

步骤 6：配置错误文档

当您为存储桶启用静态网站托管时，请输入错误文档的名称（例如，**404.html**）。为存储桶启用静态网站托管后，您可以将具有此错误文档名称的 HTML 文件上传到存储桶。

要配置错误文档

1. 创建错误文档，例如 404.html。
2. 将错误文档文件保存在本地。

   错误文档名称区分大小写，必须与启用静态网站托管时输入的名称完全匹配。例如，如果在静态网站托管对话框中为错误文档名称输入 404.html，则错误文档文件名也必须是 404.html。
3. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
4. 在存储桶 列表中，选择要用于托管静态网站的存储桶的名称。
5. 为您的存储桶启用静态网站托管，并输入错误文档的确切名称（例如 404.html）。有关更多信息，请参阅启用网站托管 和配置自定义错误文档。

   启用静态网站托管后，继续执行步骤 6。
6. 要将错误文档上传到存储桶，请执行以下操作之一：

   - 将错误文档文件拖放到控制台存储桶列表中。
   - 选择上传，然后按照提示选择并上传索引文件。

   如需分步指导，请参阅 上传对象。

步骤 7：测试您的网站端点

为存储桶配置静态网站托管后，您可以测试您的网站端点。

> **Note**
>
> Amazon S3 不支持对该网站进行 HTTPS 访问。如果要使用 HTTPS，则可以使用 Amazon CloudFront 为 Amazon S3 上托管的静态网站提供服务。
> 有关更多信息，请参阅如何使用 CloudFront 为 Amazon S3 上托管的静态网站提供服务？以及要求在查看器和 CloudFront 之间使用 HTTPS 进行通信。

1. 在 Buckets（存储桶）下，请选择存储桶的名称。
2. 请选择属性。
3. 在页面底部的 Static website hosting（静态网站托管）下，请选择 Bucket website endpoint（存储桶网站端点）。

   您的索引文档将在单独的浏览器窗口中打开。

现在，您已拥有在 Amazon S3 上托管的网站。该网站在 Amazon S3 网站端点上可用。但是，您可能有要用来从已创建的网站提供内容的域 (如 `example.com`)。您可能还需要使用 Amazon S3 的根域支持来响应对 `http://www.example.com` 和 `http://example.com` 的请求。此操作需要其他步骤。有关示例，请参阅教程：使用注册到 Route 53 的自定义域配置静态网站。

步骤 8：清除

如果您仅出于练习目的创建静态网站，则删除您所分配的 AWS 资源，使其不再产生费用。删除 AWS 资源后，您的网站将不再可用。有关更多信息，请参阅 删除存储桶。

教程：使用注册到 Route 53 的自定义域配置静态网站

假设您要在 Amazon S3 上托管静态网站。您向 Amazon Route 53 注册了一个域（如 `example.com`），希望从 Amazon S3 内容响应对 `http://www.example.com` 和 `http://example.com` 的请求。您可以使用本演练了解如何托管静态网站以及在 Amazon S3 上为具有注册到 Route 53 的自定义域名的网站创建重定向。您可以使用要在 Amazon S3 上托管的现有网站，也可以使用本演练从头开始。

完成本演练后，您可以选择使用 Amazon CloudFront 来提高网站的性能。有关更多信息，请参阅 使用 Amazon CloudFront 为网站提速。

> **Note**
>
> Amazon S3 网站端点不支持 HTTPS 或接入点。如果要使用 HTTPS，则可以使用 Amazon CloudFront 为 Amazon S3 上托管的静态网站提供服务。
> 有关如何使用 CloudFront 和 Amazon S3 安全地托管内容的教程，请参阅教程：使用 Amazon S3、Amazon CloudFront 和 Amazon Route 53 托管点播流视频。有关更多信息，请参阅如何使用 CloudFront 为 Amazon S3 上托管的静态网站提供服务？以及要求在查看器和 CloudFront 之间使用 HTTPS 进行通信。

使用 AWS CloudFormation 模板自动设置静态网站

可以使用 AWS CloudFormation 模板自动设置静态网站。AWS CloudFormation 模板设置托管安全静态网站所需的组件，这样您便能更多地关注网站内容而不是配置组件。

AWS CloudFormation 模板包含以下组件：

- Amazon S3 – 创建一个 Amazon S3 存储桶来托管静态网站。
- CloudFront – 创建 CloudFront 分配来为静态网站提速。
- Lambda@Edge – 使用 Lambda@Edge 向每个服务器响应添加安全标头。安全标头是 Web 服务器响应中的一组标头，它们告诉 Web 浏览器采取额外的安全预防措施。有关更多信息，请参阅博客文章：使用 Lambda@Edge 和 Amazon CloudFront 添加 HTTP 安全标头。

此 AWS CloudFormation 模板可供您下载和使用。有关信息和说明，请参阅《Amazon CloudFront 开发人员指南》中的安全静态网站入门。

主题

开始前的准备工作

在按照此示例中的步骤操作时，您将使用以下服务：

Amazon Route 53 – 您使用 Route 53 注册域，并定义要将您的域的 Internet 流量路由到何处。此示例介绍如何创建 Route 53 别名记录，以便将您的域（`example.com`）和子域（`www.example.com`）的流量路由到包含 HTML 文件的 Amazon S3 存储桶。

Amazon S3 – 您将使用 Amazon S3 创建存储桶、上传示例网站页面、配置权限以便每个人都可以查看内容，然后为网站托管配置存储桶。

步骤 1：将自定义域注册到 Route 53

如果您还没有已注册的域名（如 `example.com`），则可向 Route 53 注册一个域名。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的注册新域。注册您的域名后，您可以为网站托管创建并配置 Amazon S3 存储桶。

步骤 2：创建两个存储桶

要同时支持来自根域和子域的请求，您需要创建以下两个存储桶。

- 域存储桶 – `example.com`
- 子域存储桶 – `www.example.com`

这两个存储桶的名称必须与您的域名完全匹配。在此示例中，域名为 `example.com`。您将您的内容托管在根域存储桶（`example.com`）之外。您为子域存储桶（`www.example.com`）创建重定向请求。如果有人在其浏览器中输入 `www.example.com`，他们将重定向到 `example.com` 并看到该名称的 Amazon S3 存储桶中托管的内容。

为网站托管创建存储桶

以下说明概述了如何创建存储桶以用于网站托管。有关创建存储桶的详细分步说明，请参阅创建存储桶。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 选择根域存储桶：
   a. 请选择 Create bucket（创建存储桶）。
   b. 输入 Bucket name（存储桶名称）（例如 **example.com**）。
   c. 请选择要在其中创建存储桶的区域。

      请选择一个在地理上靠近您的区域可最大程度地减少延迟和成本，或满足法规要求。您选择的区域决定了您的 Amazon S3 网站端点。有关更多信息，请参阅 网站端点。
   d. 要接受默认设置并创建存储桶，请选择 Create（创建）。
3. 创建您的子域存储桶：
   a. 请选择 Create bucket（创建存储桶）。
   b. 输入 Bucket name（存储桶名称）（例如 **www.example.com**）。
   c. 请选择要在其中创建存储桶的区域。

      请选择一个在地理上靠近您的区域可最大程度地减少延迟和成本，或满足法规要求。您选择的区域决定了您的 Amazon S3 网站端点。有关更多信息，请参阅 网站端点。
   d. 要接受默认设置并创建存储桶，请选择 Create（创建）。

在下一步中，您配置 `example.com` 进行网站托管。

步骤 3：为网站托管配置根域存储桶

在此步骤中，您将根域存储桶（`example.com`）配置为网站。该存储桶将包含您的网站内容。为网站托管配置存储桶时，您可以使用网站端点访问网站。

启用静态网站托管

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择要为其启用静态网站托管的存储桶的名称。

3. 选择属性。
4. 在静态网站托管下，请选择编辑。
5. 请选择使用此存储桶托管网站。
6. 在静态网站托管下，请选择启用。
7. 在 Index document (索引文档) 中，输入索引文档的文件名，通常为 `index.html`。

   索引文档名称区分大小写，并且必须与您计划上传到 S3 存储桶的 HTML 索引文档的文件名完全匹配。当您为网站托管配置存储桶时，您必须指定索引文档。当对根域或任何子文件夹发出请求时，Amazon S3 将返回此索引文档。有关更多信息，请参阅 配置索引文档。
8. 要为 4XX 类错误提供您自己的自定义错误文档，请在错误文档中输入自定义错误文档文件名。

   错误文档名称区分大小写，并且必须与您计划上传到 S3 存储桶的 HTML 错误文档的文件名完全匹配。如果未指定自定义错误文档并发生错误，Amazon S3 返回默认 HTML 错误文档。有关更多信息，请参阅 配置自定义错误文档。
9. （可选）如果要指定高级重定向规则，请在 Redirection rules（重定向规则）中，输入 JSON 来描述规则。

   例如，您可以根据请求中的特定对象键名或前缀按条件路由请求。有关更多信息，请参阅 配置重新导向规则以使用高级条件重新导向。
10. 选择 Save changes（保存更改）。

    Amazon S3 为您的存储桶启用静态网站托管。在页面底部的静态网站托管下，您可以看到存储桶的网站端点。
11. 在静态网站托管下，记下端点。

    端点是存储桶的 Amazon S3 网站端点。将存储桶配置为静态网站后，您可以使用此端点来测试您的网站。

在编辑屏蔽公共访问权限设置并添加允许公有读取访问的存储桶策略后，您可以使用网站端点访问您的网站。

在下一步中，您将子域 (`www.example.com`) 配置为将请求重定向到您的域 (`example.com`)。

步骤 4：为网站重定向配置子域存储桶

在为网站托管配置了根域存储桶后，可以将子域存储桶配置为将所有请求重定向到该域。在此示例中，对 `www.example.com` 的所有请求都重定向到 `example.com`。

配置重定向请求

1. 在 Amazon S3 控制台上的 Buckets（存储桶）列表中，请选择您的子域存储桶名称（在本例中为 `www.example.com`）。
2. 请选择属性。
3. 在静态网站托管下，选择编辑。
4. 请选择 Redirect requests for an object（重新导向对于对象的请求）。
5. 在 Target bucket（目标存储桶）框中，输入您的根域（例如 **`example.com`**）。
6. 对于 Protocol（协议），请选择 http。
7. 选择保存更改。

步骤 5：配置网站流量的日志记录

如果想要跟踪访问您的网站的访问者人数，则可以选择对根域存储桶启用日志记录。有关更多信息，请参阅 使用服务器访问日志记录来记录请求。如果您计划使用 Amazon CloudFront 来为网站提速，也可以使用 CloudFront 日志记录。

为根域存储桶启用服务器访问日志记录

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 例如，在您创建存储桶（配置为静态网站）的同一区域中，创建用于日志记录的存储桶（例如 `logs.example.com`）。
3. 为服务器访问日志记录日志文件创建文件夹（例如，`logs`）。
4. （可选）如果要使用 CloudFront 提高网站性能，请为 CloudFront 日志文件创建一个文件夹（例如，`cdn`）。

> **Important**
>
> 当您创建或更新分配并启用 CloudFront 日志记录时，CloudFront 会更新存储桶访问控制列表（ACL），以提供 `awslogsdelivery` 账户授予将日志写入存储桶的 `FULL_CONTROL` 权限。有关更多信息，请参阅 Amazon CloudFront Developer Guide（Amazon CloudFront 开发人员指南）中的 Permissions required to configure standard logging and to access your log files（配置标准日志和访问日志文件所需的权限）。如果存储日志的存储桶使用 S3 对象所有权的强制存储桶拥有者设置来禁用 ACL，

则 CloudFront 无法将日志写入存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

5. 在 Buckets（存储桶）列表中，请选择您的根域存储桶。
6. 请选择属性。
7. 在 Server access logging（服务器访问日志记录）下，请选择 Edit（编辑）。
8. 请选择 Enable。
9. 在 Target bucket（目标存储桶）下，请选择服务器访问日志的存储桶和文件夹目标：
   - 浏览到文件夹和存储桶位置：
     1. 请选择 Browse S3（浏览 S3）。
     2. 请选择存储桶名称，然后选择日志文件夹。
     3. 请选择 Choose path（选择路径）。
   - 输入 S3 存储桶路径，例如 `s3://logs.example.com/logs/`。
10. 选择保存更改。

    在日志存储桶中，您现在可以访问日志。Amazon S3 每 2 小时将网站访问日志写入您的日志存储桶一次。

步骤 6：上传索引和网站内容

在此步骤中，您可以将索引文档和可选网站内容上传到您的根域存储桶。

当您为存储桶启用静态网站托管时，您可以输入索引文档的名称（例如，**`index.html`**）。为存储桶启用静态网站托管后，您可以将具有此索引文档名称的 HTML 文件上传到存储桶。

配置索引文档

1. 创建 `index.html` 文件。

   如果您没有 `index.html` 文件，则可以使用以下 HTML 创建一个：

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>My Website Home Page</title>
</head>
<body>
```

```
  <h1>Welcome to my website</h1>
  <p>Now hosted on Amazon S3!</p>
</body>
</html>
```

2. 将索引文件保存在本地。

   索引文档文件名必须与您在静态网站托管对话框中输入的索引文档名称完全匹配。索引文档名称区分大小写。例如，如果在静态网站托管对话框中为索引文档名称输入 `index.html`，则索引文档文件名也必须是 `index.html`，而不是 `Index.html`。

3. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

4. 在存储桶列表中，请选择要用于托管静态网站的存储桶的名称。

5. 为您的存储桶启用静态网站托管，并输入索引文档的确切名称（例如 `index.html`）。有关更多信息，请参阅 启用网站托管。

   启用静态网站托管后，继续执行步骤 6。

6. 要将索引文档上传到存储桶，请执行以下操作之一：

   - 将索引文件拖放到控制台存储桶列表中。
   - 选择上传，然后按照提示选择并上传索引文件。

   如需分步指导，请参阅 上传对象。

7. （可选）将其他网站内容上传到您的存储桶。

步骤 7：上传错误文档

当您为存储桶启用静态网站托管时，请输入错误文档的名称（例如，**`404.html`**）。为存储桶启用静态网站托管后，您可以将具有此错误文档名称的 HTML 文件上传到存储桶。

要配置错误文档

1. 创建错误文档，例如 `404.html`。

2. 将错误文档文件保存在本地。

   错误文档名称区分大小写，必须与启用静态网站托管时输入的名称完全匹配。例如，如果在静态网站托管对话框中为错误文档名称输入 `404.html`，则错误文档文件名也必须是 `404.html`。

3. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
4. 在存储桶 列表中，选择要用于托管静态网站的存储桶的名称。
5. 为您的存储桶启用静态网站托管，并输入错误文档的确切名称（例如 `404.html`）。有关更多信息，请参阅启用网站托管 和配置自定义错误文档。

   启用静态网站托管后，继续执行步骤 6。
6. 要将错误文档上传到存储桶，请执行以下操作之一：

   - 将错误文档文件拖放到控制台存储桶列表中。
   - 选择上传，然后按照提示选择并上传索引文件。

   如需分步指导，请参阅 上传对象。

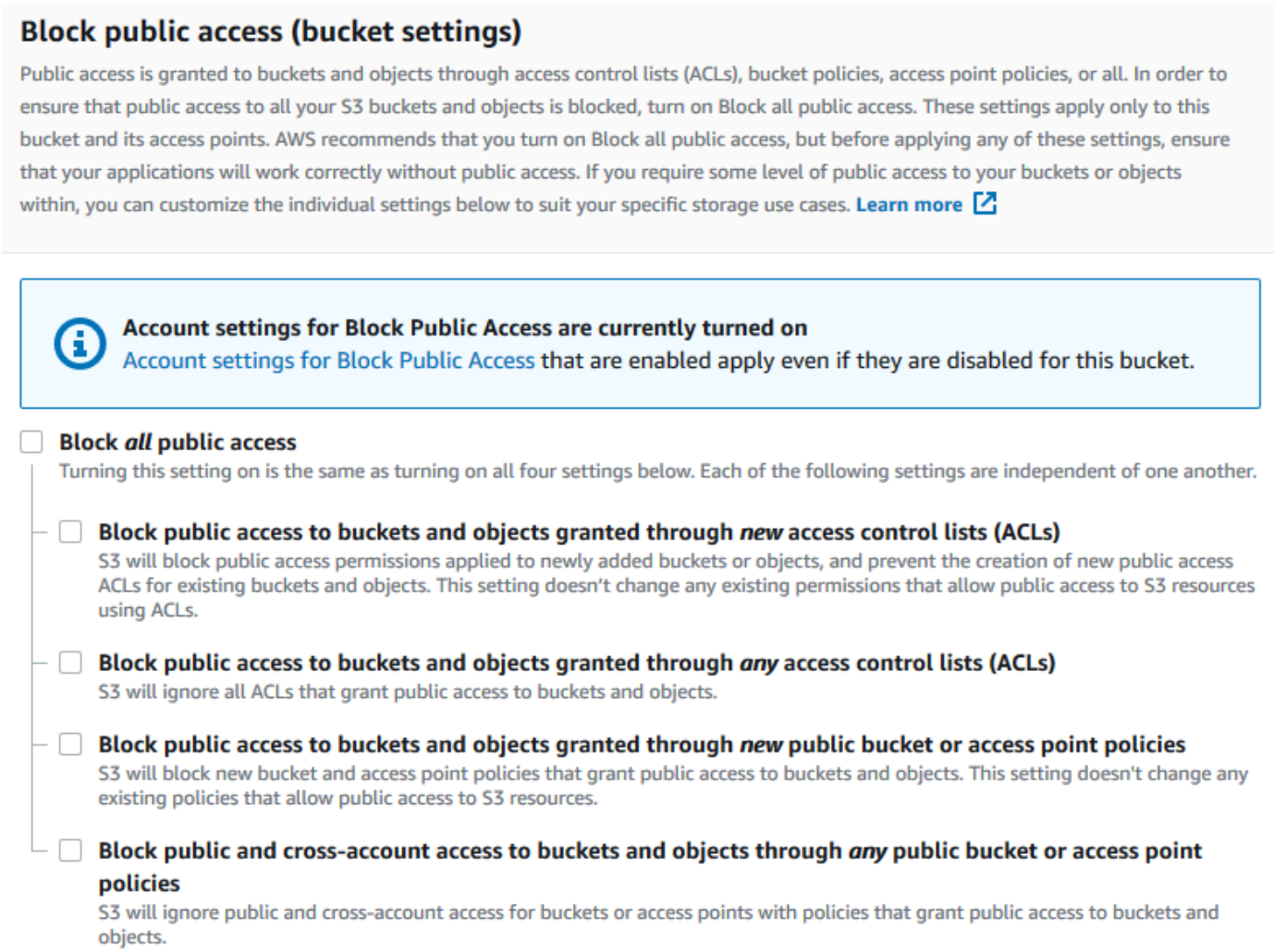
步骤 8：编辑 S3 屏蔽公共访问权限设置

在此示例中，您编辑域存储桶（`example.com`）的屏蔽公共访问权限设置以允许公有访问。

默认情况下，Amazon S3 阻止对您的账户和存储桶的公有访问权限。如果要使用存储桶托管静态网站，您可以使用以下步骤编辑您的屏蔽公共访问权限设置。

> **Warning**
>
> 在完成这些步骤之前，请查看阻止对您的 Amazon S3 存储的公有访问，来确保您了解并接受支持公共访问权限所涉及的风险。当您关闭屏蔽公共访问权限设置以使您的存储桶变为公有时，Internet 上的任何人都可以访问您的存储桶。我们建议您阻止对存储桶的所有公有访问。

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 请选择已配置为静态网站的存储桶的名称。
3. 选择权限。
4. 在屏蔽公共访问权限（存储桶设置）下，请选择编辑。
5. 清除阻止所有公有访问，然后选择保存更改。



Amazon S3 关闭了存储桶的屏蔽公共访问权限设置。要创建公有静态网站，可能还必须为您的账户配置屏蔽公共访问权限设置，然后再添加存储桶策略。如果当前已开启账户的屏蔽公共访问权限设置，您将在屏蔽公共访问权限（存储桶设置）下看到一条备注。

步骤 9：附加存储桶策略

在此示例中，您将存储桶策略附加到域存储桶（`example.com`）以允许进行公有读取访问。例如，您可以将示例存储桶策略中的 *`Bucket-Name`* 替换为域存储桶的名称，例如 `example.com`。

在编辑 S3 屏蔽公共访问权限设置后，您可以添加存储桶策略以授予对存储桶的公有读取访问权限。当您授予公有读取访问权限时，Internet 上的任何人都可以访问您的存储桶。

> **Important**
>
> 下面的策略仅供举例说明，仍允许完全访问您存储桶的内容。在继续执行此步骤之前，请查看如何保护 Amazon S3 存储桶中的文件？，以确保您了解保护 S3 存储桶中文件的最佳实践以及授予公有访问权限所涉及的风险。

1. 在存储桶 下，请选择存储桶的名称。
2. 选择权限。
3. 在存储桶策略下，请选择编辑。
4. 要授予对网站的公有读取访问权限，请复制以下存储桶策略，将其粘贴到存储桶策略编辑器中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. 将 `Resource` 更新为您的存储桶名称。

   在上述示例存储桶策略中，*Bucket-Name* 是存储桶名称的占位符。要将此存储桶策略用于您自己的存储桶，您必须更新此名称以匹配您的存储桶名称。

6. 选择保存更改。

   此时将显示一条消息，指示存储桶策略已成功添加。

   如果您看到显示 `Policy has invalid resource` 的错误，请确认存储桶策略中的存储桶名称与您的存储桶名称匹配。有关添加存储桶策略的信息，请参阅如何添加 S3 存储桶策略？

如果您收到错误消息且无法保存存储桶策略，请检查您的账户和存储桶的屏蔽公共访问权限设置以确认您允许对存储桶进行公有访问。

在下一步中，您可以找出您的网站端点并测试您的域端点。

步骤 10：测试您的域端点

在将您的域存储桶配置为托管公有网站之后，可以测试您的端点。有关更多信息，请参阅 网站端点。您只能测试域存储桶的端点，因为您的子域存储桶是为网站重定向设置的，而不是静态网络托管。

> **Note**
>
> Amazon S3 不支持对该网站进行 HTTPS 访问。如果要使用 HTTPS，则可以使用 Amazon CloudFront 为 Amazon S3 上托管的静态网站提供服务。
> 有关更多信息，请参阅如何使用 CloudFront 为 Amazon S3 上托管的静态网站提供服务？以及要求在查看器和 CloudFront 之间使用 HTTPS 进行通信。

1. 在 Buckets（存储桶）下，请选择存储桶的名称。
2. 请选择属性。
3. 在页面底部的 Static website hosting（静态网站托管）下，请选择 Bucket website endpoint（存储桶网站端点）。

   您的索引文档将在单独的浏览器窗口中打开。

接下来，您使用 Amazon Route 53 使客户能够使用这两个自定义 URL 导航到您的站点。

步骤 11：为您的域和子域添加别名记录

在此步骤中，您创建别名记录并将其添加到您的域的托管区域，而这些别名记录将映射 `example.com` 和 `www.example.com`。别名记录使用 Amazon S3 网站端点，而不使用 IP 地址。Amazon Route 53 保留了别名记录和 Amazon S3 存储桶所在的 IP 地址之间的映射。您创建两个别名记录，一个用于根域，一个用于子域。

为根域和子域添加别名记录

为您的根域 (**example.com**) 添加别名记录

1. 通过以下网址打开 Route 53 控制台：https://console.aws.amazon.com/route53/。

   > **Note**
   >
   > 如果您尚未使用 Route 53，请参阅 Amazon Route 53 开发人员指南中的步骤 1：注册域。完成设置后，可继续按说明操作。

2. 请选择托管区域。
3. 在托管区域列表中，请选择与您的域名匹配的托管区域的名称。
4. 请选择创建记录。
5. 请选择切换到向导。

   > **Note**
   >
   > 如果要使用快速创建来创建别名记录，请参阅配置 Route 53 以将流量路由到 S3 存储桶。

6. 请选择简单路由，然后选择下一步。
7. 请选择定义简单记录。
8. 在记录名称中，接受默认值，该值为您的托管区域和您的域的名称。
9. 在值/流量路由至中，请选择向 S3 网站端点添加别名。
10. 请选择区域。
11. 请选择 S3 存储桶。

    存储桶名称应与 Name（名称）框中显示的名称相匹配。例如，在选择 S3 存储桶列表中，存储桶名称将与创建存储桶所在区域的 Amazon S3 网站端点一起显示 `s3-website-us-west-1.amazonaws.com (example.com)`。

    如果出现以下情况，请选择 S3 存储桶将列出一个存储桶：

    - 您已将存储桶配置为静态网站。
    - 存储桶名称与您正在创建的记录的名称相同。
    - 当前 AWS 账户 创建了存储桶。

如果您的存储桶未显示在选择 S3 存储桶列表中，请输入在其中创建存储桶的区域的 Amazon S3 网站端点，例如 **s3-website-us-west-2.amazonaws.com**。有关 Amazon S3 网站端点的完整列表，请参阅 Amazon S3 网站端点。有关别名目标的更多信息，请参阅 Amazon Route 53 开发人员指南中的 值/路由流量至。

12. 在 Record type (记录类型) 中，选择 A – Routes traffic to an IPv4 address and some AWS resources (A – 将流量路由到 IPv4 地址和某些 AWS 资源)。
13. 对于评估目标运行状况，请选择否。
14. 请选择定义简单记录。

为您的子域添加别名记录 (**www.example.com**)

1. 在配置记录下，请选择定义简单记录。
2. 在子域的记录名称中，键入 www。
3. 在值/流量路由至中，请选择向 S3 网站端点添加别名。
4. 请选择区域。
5. 请选择 S3 存储桶，例如 s3-website-us-west-2.amazonaws.com (www.example.com)。

   如果您的存储桶未显示在选择 S3 存储桶列表中，请输入在其中创建存储桶的区域的 Amazon S3 网站端点，例如 **s3-website-us-west-2.amazonaws.com**。有关 Amazon S3 网站端点的完整列表，请参阅 Amazon S3 网站端点。有关别名目标的更多信息，请参阅 Amazon Route 53 开发人员指南中的 值/路由流量至。
6. 在 Record type (记录类型) 中，选择 A – Routes traffic to an IPv4 address and some AWS resources (A – 将流量路由到 IPv4 地址和某些 AWS 资源)。
7. 对于评估目标运行状况，请选择否。
8. 请选择定义简单记录。
9. 在配置记录页面上，请选择创建记录。

> **Note**
>
> 更改通常在 60 秒内传播到所有 Route 53 服务器。传播完成后，您可以使用在此步骤中创建的别名记录的名称将流量路由到 Amazon S3 存储桶。

为根域和子域添加别名记录（旧 Route 53 控制台）

为您的根域 (**example.com**) 添加别名记录

Route 53 控制台进行了重新设计。在 Route 53 控制台中，您可以暂时使用旧控制台。如果您选择使用旧 Route 53 控制台，请使用以下过程。

1. 通过以下网址打开 Route 53 控制台：https://console.aws.amazon.com/route53/。

   > **Note**
   >
   > 如果您尚未使用 Route 53，请参阅 Amazon Route 53 开发人员指南中的步骤 1：注册域。完成设置后，可继续按说明操作。

2. 请选择 Hosted Zones (托管区域)。
3. 在托管区域列表中，请选择与您的域名匹配的托管区域的名称。
4. 请选择 Create Record Set。
5. 指定以下值：

   名称

   接受默认值，该值为您的托管区域和您的域的名称。

   对于根域，您不需要在 Name (名称) 字段中输入任何其他信息。

   类型

   请选择 A – IPv4 address（A – IPv4 地址）。

   别名

   请选择是。

   别名目标

   在列表的 S3 website endpoints（S3 网站端点）部分，请选择您的存储桶名称。

   存储桶名称应与 Name（名称）框中显示的名称相匹配。在 Alias Target（别名目标）列表中，存储桶名称后跟在其中创建存储桶的区域的 Amazon S3 网站端点，例如 `example.com (s3-website-us-west-2.amazonaws.com)`。Alias Target（别名目标）会在以下情况下列出存储桶：

   - 您已将存储桶配置为静态网站。

- 存储桶名称与您正在创建的记录的名称相同。
- 当前 AWS 账户 创建了存储桶。

如果您的存储桶未显示在 Alias Target（别名目标）列表中，请输入在其中创建存储桶的区域的 Amazon S3 网站端点，例如 `s3-website-us-west-2`。有关 Amazon S3 网站端点的完整列表，请参阅 Amazon S3 网站端点。有关别名目标的更多信息，请参阅 Amazon Route 53 开发人员指南中的 值/路由流量至。

路由策略

接受默认值 Simple。

Evaluate Target Health

接受默认值 No。

6. 请选择创建。

为您的子域添加别名记录 (**`www.example.com`**)

1. 在您的根域 (`example.com`) 的托管区域中，请选择 Create Record Set (创建记录集)。
2. 指定以下值：

名称

对于子域，在框中输入 `www`。

类型

请选择 A – IPv4 address（A – IPv4 地址）。

别名

请选择是。

别名目标

在列表的 S3 website endpoints（S3 网站端点）部分中，请选择 Name（名称）字段中显示的相同存储桶名称，例如 `www.example.com (s3-website-us-west-2.amazonaws.com)`。

路由策略

接受默认值 Simple。

Evaluate Target Health

接受默认值 No。

3. 请选择创建。

> **Note**
>
> 更改通常在 60 秒内传播到所有 Route 53 服务器。传播完成后，您可以使用在此步骤中创建的别名记录的名称将流量路由到 Amazon S3 存储桶。

步骤 12：测试网站

验证网站和重定向正常工作。在浏览器中，输入 URL。在本示例中，可以尝试以下 URL：

- 域（`http://example.com`）– 显示 `example.com` 存储桶中的索引文档。
- 子域 (`http://www.example.com`) – 将您的请求重定向到 `http://example.com`。您会在 `example.com` 存储桶中看到索引文档。

如果您的网站或重定向链接不起作用，则可以尝试以下操作：

- 清除缓存 – 清除 Web 浏览器缓存。
- 检查名称服务器 – 如果在清除缓存后，您的网页和重定向链接无法正常工作，则可以比较域的名称服务器和托管区域的名称服务器。如果名称服务器不匹配，您可能需要更新域名服务器，以便匹配在托管区域下方列出的服务器。有关更多信息，请参阅为域添加或更改名称服务器和粘附记录。

在成功测试根域和子域后，您可以设置 Amazon CloudFront 分配以提高网站的性能，并提供可用于查看网站流量的日志。有关更多信息，请参阅 使用 Amazon CloudFront 为网站提速。

使用 Amazon CloudFront 为网站提速

您可以使用 Amazon CloudFront 来提高 Amazon S3 网站的性能。CloudFront 可将您的网站文件（如 HTML、图像和视频）提供给全球各地的数据中心（称为边缘站点）使用。当访问者从您的网站请求文件时，CloudFront 会自动将请求重定向到最近边缘站点上的文件副本。这时的下载速度会比访问者从更远的数据中心请求该内容更快。

CloudFront 会将内容在边缘站点上缓存您指定的时间。如果访问者请求已过期的缓存内容，CloudFront 会检查来源服务器，确定该内容是否有更新的版本可用。如果有更新的版本，则 CloudFront 将新版本复制到该边缘站点。当访问者请求内容时，您对原始内容所做的更改便会复制到边缘站点。

在没有 Route 53 的情况下使用 CloudFront

本页面上的教程使用 Route 53 来指向您的 CloudFront 分配。但是，如果您想在不使用 Route 53 的情况下使用 CloudFront 提供托管在 Amazon S3 存储桶中的内容，请参阅 Amazon CloudFront Tutorials: Setting up a Dynamic Content Distribution for Amazon S3。在使用 CloudFront 提供托管在 Amazon S3 存储桶中的内容时，您可以使用任何存储桶名称，并且同时支持 HTTP 和 HTTPS。

使用 AWS CloudFormation 模板自动执行设置过程

有关使用 AWS CloudFormation 模板配置安全静态网站以创建 CloudFront 分配来为您的网站提供服务的更多信息，请参阅《Amazon CloudFront 开发人员指南》中的安全静态网站入门。

主题

步骤 1：创建 CloudFront 分配

首先，创建 CloudFront 分配。这将使您的网站可供全球各地的数据中心使用。

使用 Amazon S3 源创建分配

1. 通过 打开 CloudFront 控制台https://console.aws.amazon.com/cloudfront/v4/home
2. 选择 Create Distribution（创建分配）。
3. 在 Create Distribution（创建分配）页面上的 Origin Settings（源设置）部分中，对于 Origin Domain Name（源域名），输入您的存储桶的 Amazon S3 网站端点，例如 **`example.com.s3-website.us-west-1.amazonaws.com`**。

   CloudFront 将为您填写 Origin ID (源 ID)。
4. 对于 Default Cache Behavior Settings (默认缓存行为设置)，将值保留设置为默认值。

   当为查看器协议策略使用默认设置时，您可以为静态网站使用 HTTPS。有关这些配置选项的更多信息，请参阅《Amazon CloudFront 开发人员指南》中的您创建或更新 Web 分配时指定的值。

5. 对于 Distribution Settings，执行以下操作：

   a. 将 Price Class (价格级别) 的设置保留为 Use All Edge Locations (Best Performance) (使用所有节点 (最佳性能))。

   b. 将 Alternate Domain Names (CNAMEs) (备用域名(CNAME)) 设置为根域和 www 子域。在本教程中，它们是 `example.com` 和 `www.example.com`。

   > **Important**
   >
   > 在执行该步骤之前，请注意使用备用域名的要求，尤其是需要使用有效的 SSL/TLS 证书。

   c. 对于 SSL 证书，选择自定义 SSL 证书 (example.com)，然后选择涵盖域名和子域名的自定义证书。

   有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的 SSL 证书。

   d. 在 Default Root Object (默认根对象) 中，输入索引文档的名称，例如 `index.html`。

   如果用于访问分配的 URL 不包含文件名，CloudFront 分配将返回索引文档。Default Root Object (默认根对象) 应该与静态网站的索引文档的名称完全匹配。有关更多信息，请参阅 配置索引文档。

   e. 将 Logging (日志记录) 设置为 On (打开)。

   > **Important**
   >
   > 当您创建或更新分配并启用 CloudFront 日志记录时，CloudFront 会更新存储桶访问控制列表 (ACL) ，以提供 `awslogsdelivery` 账户 `FULL_CONTROL` 权限，从而将日志写入您的存储桶。有关更多信息，请参阅 Amazon CloudFront Developer Guide (Amazon CloudFront 开发人员指南) 中的 Permissions required to configure standard logging and to access your log files (配置标准日志和访问日志文件所需的权限) 。如果存储日志的存储桶使用 S3 对象所有权的强制存储桶拥有者设置来禁用 ACL，则 CloudFront 无法将日志写入存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。 。

   f. 对于 Bucket for Logs，选择您创建的日志记录存储桶。

   有关配置日志记录存储桶的更多信息，请参阅 (可选) 记录 Web 流量。

g. 如果您要将由流量生成的日志存储到文件夹中的 CloudFront 分配，请在 Log Prefix (日志前缀) 中键入文件夹名称。

h. 将所有其他设置保留为默认值。

6. 选择 Create Distribution。

7. 要查看分配的状态，请在控制台中找到该分配，然后检查 Status 列。

   `InProgress` (进行中) 状态表示分配尚未完成部署。

   分配部署完毕后，您可以使用新的 CloudFront 域名来引用您的内容。

8. 记录 CloudFront 控制台中显示的 Domain Name (域名) 值，例如 `dj4p1rv6mvubz.cloudfront.net`。

9. 要验证您的 CloudFront 分配是否正常运行，请在 Web 浏览器中输入该分配的域名。

   如果您的网站是可见的，则 CloudFront 分配正常工作。如果您的网站具有向 Amazon Route 53 注册的自定义域，您将需要 CloudFront 域名来在下一步更新记录集。

步骤 2：更新域和子域的记录集

现在您已成功地创建 CloudFront 分配，请更新 Route 53 中的别名记录以指向新的 CloudFront 分配。

更新别名记录以指向 CloudFront 分配

1. 通过以下网址打开 Route 53 控制台：https://console.aws.amazon.com/route53/。

2. 在左侧导航中，选择 Hosted zones（托管区）。

3. 在 Hosted Zones（托管区）页上，选择为您的子域创建的托管区域，例如 `www.example.com`。

4. 在 Records（记录）下，选择您为子域创建的 A 记录。

5. 在 Record details（记录详细信息）下，选择 Edit record（编辑记录）。

6. 在 Route traffic to（将流量路由至）下，选择 Alias to CloudFront distribution（别名到 CloudFront 分配）。

7. 在 Choose distribution（选择分配）下，选择 CloudFront 分配。

8. 选择 Save。

9. 要将根域的 A 记录重定向到 CloudFront 分配，请对根域重复此过程，例如 `example.com`。

   记录集的更新需要 2 – 48 小时才能生效。

10. 要查看新的 A 记录是否已生效，请在 Web 浏览器中输入您的子域 URL，例如 `http://www.example.com`。

    如果您的浏览器不再重定向至根域（例如，`http://example.com`），则说明新的 A 记录已生效。当新 A 记录生效时，由新 A 记录路由到 CloudFront 分配的流量不会重定向到根域。任何使用 `http://example.com` 或 `http://www.example.com` 引用站点的访问者都会重定向到最近的 CloudFront 边缘站点，在这里可体验更快速的下载。

    > **Tip**
    >
    > 浏览器可以缓存重定向设置。如果您认为新的 A 记录设置应该已经生效，但是您的浏览器仍然将 `http://www.example.com` 重定向至 `http://example.com`，请尝试清除浏览器的历史记录和缓存，然后关闭再重新打开浏览器应用程序，或使用其他 Web 浏览器。

（可选）步骤 3：检查日志文件

访问日志会告诉您有多少人正在访问网站。它们还包含有价值的业务数据，您可以使用 Amazon EMR 等其他服务分析这些数据。

CloudFront 日志存储在您在创建 CloudFront 分配并启用日志记录时所选择的存储桶和文件夹中。CloudFront 将在相应请求提出后的 24 小时内将日志写入您的日志存储桶。

查看网站的日志文件

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 为网站选择日志记录存储桶的名称。
3. 选择 CloudFront 日志文件夹。
4. 先下载由 CloudFront 编写的 `.gzip` 文件，然后再打开这些文件。

   如果您仅出于练习目的创建网站，则可以删除您所分配的资源，使其不再产生费用。为此，请参阅清理示例资源。删除 AWS 资源后，您的网站将不再可用。

清理示例资源

如果您出于学习目的创建静态网站，应删除您分配的AWS资源，使其不再产生费用。删除 AWS 资源后，您的网站将不再可用。

任务

- 步骤 1：删除 Amazon CloudFront 分配
- 步骤 2：删除 Route 53 托管区域
- 步骤 3：禁用日志记录并删除 S3 存储桶

步骤 1：删除 Amazon CloudFront 分配

在删除 Amazon CloudFront 分配之前，您必须先将其禁用。已禁用的分发不再起作用，并且不会产生费用。您可以随时启用已禁用的分发。已禁用的分发在删除后将不再可用。

禁用并删除 CloudFront 分配

1. 通过 打开 CloudFront 控制台https://console.aws.amazon.com/cloudfront/v4/home
2. 选择要禁用的分配，然后选择禁用。
3. 当系统提示确认时，选择是，禁用。
4. 选择禁用的分配，然后选择删除。
5. 当系统提示进行确认时，选择 Yes, Delete。

步骤 2：删除 Route 53 托管区域

在删除托管区域之前，您必须先删除已创建的记录集。您不需要删除 NS 和 SOA 记录；删除托管区域时，会自动删除这些记录。

删除记录集

1. 通过以下网址打开 Route 53 控制台：https://console.aws.amazon.com/route53/。
2. 在域名列表中，选择您的域名，然后选择 Go to Record Sets。
3. 在记录集列表中，选择您创建的 A 记录。

   每个记录集的类型均列在 Type (类型) 列中。
4. 选择 Delete Record Set。
5. 当系统提示进行确认时，选择 Confirm。

删除 Route 53 托管区域

1. 紧接上一个步骤，选择 Back to Hosted Zones。

2. 选择您的域名，然后选择 Delete Hosted Zone。
3. 当系统提示进行确认时，选择 Confirm。

步骤 3：禁用日志记录并删除 S3 存储桶

在删除您的 S3 存储桶之前，请确保已禁用该存储桶的日志记录。否则，在您删除存储桶时，AWS会继续将日志写入其中。

禁用存储桶的日志记录

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）下，选择存储桶名称，然后选择 Properties（属性）。
3. 从 Properties 中选择 Logging。
4. 取消选中启用复选框。
5. 选择保存。

现在您可以删除存储桶。有关更多信息，请参阅 删除存储桶。

创建、配置和使用 Amazon S3 存储桶

要将数据存储在 Amazon S3 中，您需要使用称为存储桶和对象的资源。存储桶是对象的容器。对象指的是一个文件和描述该文件的任何元数据。

要将对象存储在 Amazon S3 中，您需要创建存储桶，然后将该对象上传到存储桶。当对象位于存储桶时，您可以将其打开、下载并移动它。当您不再需要对象或存储桶时，可以清理您的资源。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

> **Note**
>
> 使用 Amazon S3，您只需按实际用量付费。有关 Amazon S3 特征和定价的更多信息，请参阅 Amazon S3。如果您是 Amazon S3 的新客户，则可以免费开始使用 Amazon S3。有关更多信息，请参阅 AWS 免费套餐。

本节中的主题概述了在 Amazon S3 中使用存储桶。它们包括有关命名、创建、访问和删除存储桶的信息。有关查看或列出存储桶中对象的更多信息，请参阅 组织、列出和处理对象。

主题

- 使用申请方付款存储桶进行存储传输和使用
- 存储桶配额、限制和局限性

存储桶概述

要向 Amazon S3 上传数据（照片、视频、文档等），您必须首先在其中一个 AWS 区域 中创建 S3 存储桶。

存储桶是 Amazon S3 中用于存储对象的容器。您可以在存储桶中存储任意数量的对象，并且账户中最多可以有 100 个存储桶。要请求提高限额，请访问服务限额控制台。

每个对象都储存在一个存储桶中。例如，如果名为 photos/puppy.jpg 的对象存储在美国西部（俄勒冈州）区域的 amzn-s3-demo-bucket 存储桶中，则可使用 URL https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg 对该对象进行寻址。有关更多信息，请参阅访问存储桶。

对于实施，存储桶和对象是 AWS 资源，而 Amazon S3 提供 API 供您管理资源。例如，您可以使用 Amazon S3 API 创建存储桶并上传对象。您还可以使用 Amazon S3 控制台执行这些操作。该控制台使用 Amazon S3 API 将请求发送到 Amazon S3。

本节介绍如何使用存储桶。有关使用对象的信息，请参阅Amazon S3 对象概述。

Amazon S3 支持全局存储桶，这意味着每个存储桶名称在分区内的所有AWS 区域中所有AWS 账户内都必须是唯一的。分区是一组区域。AWS 目前有三个分区：aws（标准区域）、aws-cn（中国区域）和 aws-us-gov（AWS GovCloud (US)）。

在创建存储桶之后，该存储桶的名称不能被同一分区中的另一个 AWS 账户 使用，直到存储桶被删除。不应依赖特定的存储桶命名约定来实现可用性或安全验证。有关存储桶命名指南，请参阅存储桶命名规则。

Amazon S3 在指定的区域中创建存储桶。为减少延迟、最大限度地降低成本或满足法规要求，请选择地理上靠近您的任何 AWS 区域。例如，如果您位于欧洲，您可能会发现在欧洲（爱尔兰）或欧洲（法兰克福）区域创建存储桶十分有利。有关 Amazon S3 区域的列表，请参阅《AWS 一般参考》中的区域和端点。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

> **Note**
>
> 属于您在特定 AWS 区域 中创建的存储桶的对象永远不会离开该区域，除非您明确将其传输到另一个区域。例如，在欧洲（爱尔兰）区域存储的对象将一直留在该区域。

主题

关于权限

您可以使用 AWS 账户根用户 凭证创建存储桶并执行任何其他 Amazon S3 操作。但是，我们建议您不要使用 AWS 账户 的根用户凭证发出请求，例如创建存储桶。而是创建一个 AWS Identity and Access Management (IAM) 用户，并向该用户授予完全访问权限 (用户在默认情况下没有任何权限)。

这些用户称为管理员。您可以使用管理员用户凭证而不是您账户的根用户凭证来与 AWS 交互和执行任务，例如创建存储桶，创建用户和授予他们权限。

有关更多信息，请参阅《AWS 一般参考》中的 AWS 账户根用户 凭证和 IAM 用户凭证以及《IAM 用户指南》中的 IAM 中的安全最佳实践。

创建资源的 AWS 账户 拥有该资源。例如，如果您在 AWS 账户 中创建一个 IAM 用户并向该用户授予创建存储桶的权限，则该用户可以创建存储桶。但该用户不拥有存储桶；该存储桶属于用户所属的 AWS 账户 所有。用户需要资源拥有者提供其他权限来执行任何其他存储桶操作。有关为您的 Amazon S3 资源管理权限的更多信息，请参阅Amazon S3 的身份和访问管理。

管理对存储桶的公有访问

通过存储桶策略和/或访问控制列表（ACL）授予对存储桶和对象的公共访问权限。为了帮助您管理对 Amazon S3 资源的公有访问，Amazon S3 提供屏蔽公共访问权限的设置。Amazon S3 屏蔽公共访问权限设置可以覆盖 ACL 和存储桶策略，以便您可以对这些资源的公有访问实施统一限制。您可以将屏蔽公共访问权限设置应用于您的账户中的单个存储桶或所有存储桶。

为确保您的所有 Amazon S3 存储桶和对象都屏蔽了公共访问权限，在创建新存储桶时，默认情况下会启用“屏蔽公共访问权限”的所有四个设置。我们建议也为您的账户启用“屏蔽公共访问权限”的所有四个设置。这些设置阻止所有当前和将来的存储桶的所有公有访问。

在应用这些设置之前，请确认您的应用程序在没有公有访问的情况下能够正常工作。如果您需要对存储桶或对象进行某种级别的公共访问，例如，要按照使用 Amazon S3 托管静态网站中所述托管静态网站，则可以自定义各个设置以符合您的存储使用案例要求。有关更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

但是，我们建议将“屏蔽公共访问权限”保留为启用状态。如果要将所有四个“屏蔽公共访问权限”设置保持为启用状态并托管静态网站，则可以使用 Amazon CloudFront 来源访问控制（OAC）。Amazon CloudFront 提供了设置安全静态网站所需的功能。Amazon S3 静态网站仅支持 HTTP 端点。Amazon CloudFront 使用 Amazon S3 的持久存储，同时提供额外的安全标头，如 HTTPS。HTTPS 通过加密正常 HTTP 请求并防范常见的网络攻击来增强安全性。

有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的安全静态网站入门。

> **Note**
>
> 如果您在列出存储桶及其公有访问设置时看到 `Error`，则您可能不具备所需的权限。确保您已将以下权限添加到您的用户或角色策略：
>
> ```
> s3:GetAccountPublicAccessBlock
> s3:GetBucketPublicAccessBlock
> s3:GetBucketPolicyStatus
> s3:GetBucketLocation
> s3:GetBucketAcl
> s3:ListAccessPoints
> s3:ListAllMyBuckets
> ```
>
> 在极少数情况下，请求也可能因 AWS 区域中断而失败。

存储桶配置选项

Amazon S3 支持各种供您用于配置存储桶的选项。例如，您可以配置存储桶用于网站托管、添加配置以管理存储桶中对象的生命周期以及配置存储桶以记录对存储桶的所有访问。Amazon S3 支持供您存储和管理存储桶配置信息的子资源。您可以使用 Amazon S3 API 创建和管理这些子资源。但是，还可以使用控制台或 AWS SDK。

> **Note**
>
> 还有对象级配置。例如，可以通过特定于该对象的配置访问控制列表 (ACL) 来配置对象级权限。

这些称为子资源，因为它们存在于特定存储桶或对象的上下文中。下表列出使您可以管理特定于存储桶的配置的子资源。

子资源	描述
cors（跨源资源共享）	您可以配置存储桶以便允许跨源请求。 有关更多信息，请参阅 使用跨源资源共享 (CORS)。
事件通知	您可以使存储桶向您发送特定存储桶事件的通知。 有关更多信息，请参阅 Amazon S3 事件通知。
生命周期	您可以为存储桶中明确定义了生命周期的对象定义生命周期规则。例如，您可以定义一个规则以在创建一年之后存档对象，或在创建 10 年之后删除对象。 有关更多信息，请参阅 管理存储生命周期。
位置	创建存储桶时，请指定需要 Amazon S3 在其中创建存储桶的 AWS 区域。Amazon S3 将此信息存储在位置子资源中并提供 API 用于检索此信息。
日志记录	日志记录使您可以跟踪针对存储桶的访问请求。每个访问日志记录都提供有关单个访问请求的详细信息，如请求者、存储桶名称、请求时间、请求操作、响应状态和错误代码（如果有）。访问日志信息可能在安全和访问审核方面十分有用。它还可以帮助您了解您的客户群并了解您的 Amazon S3 账单。 有关更多信息，请参阅 使用服务器访问日志记录来记录请求。
对象锁定	要使用 S3 对象锁定，您必须为存储桶启用它。您还可以选择配置将应用于存储桶中放置的新对象的默认保留模式和保留期限。 有关更多信息，请参阅 使用 S3 对象锁定。

子资源	描述
策略 和 ACL（访问控制列表）	所有资源（如存储桶和对象）在默认情况下为私有。Amazon S3 支持存储桶策略和访问控制列表（ACL）选项，供您用于授予和管理存储桶级权限。Amazon S3 将权限信息存储在策略 和 acl 子资源中。 有关更多信息，请参阅 Amazon S3 的身份和访问管理。
复制	复制是在相同或跨不同 AWS 区域 中的存储桶自动、异步地复制对象。有关更多信息，请参阅 复制对象概述。
requestPayment	默认情况下，创建存储桶的 AWS 账户（存储桶拥有者）支付从存储桶进行的下载。存储桶拥有者可以使用此子资源指定对请求下载的人员收取下载费用。Amazon S3 提供了一个 API，用于管理此子资源。 有关更多信息，请参阅 使用申请方付款存储桶进行存储传输和使用。
标记	您可以向存储桶添加成本分配标签以便对 AWS 成本进行分类和跟踪。Amazon S3 提供标记 子资源以对存储桶上的标签进行存储和管理。通过使用应用于您存储桶的标签，AWS 可生成成本分配报告，其中按标签汇总了使用情况和成本。 有关更多信息，请参阅 Amazon S3 的账单和使用情况报告。
传输加速	Transfer Acceleration 可在您的客户端与 S3 存储桶之间实现快速、轻松、安全的远距离文件传输。Transfer Acceleration 利用 Amazon CloudFront 中的全球分布式边缘站点。 有关更多信息，请参阅 使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输。
版本控制	版本控制可帮助恢复意外覆盖和删除。 我们建议将版本控制作为从错误删除或覆盖恢复对象的最佳实践。 有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

子资源	描述
网站	您可以配置存储桶以便用于静态网站托管。Amazon S3 通过创建网站 子资源来存储此配置。 有关更多信息，请参阅 使用 Amazon S3 托管静态网站。

存储桶命名规则

以下规则适用于在 Amazon S3 中命名通用存储桶和目录存储桶：

主题

通用存储桶命名规则

以下命名规则适用于通用存储桶。

- 存储桶名称必须介于 3（最少）到 63（最多）个字符之间。
- 存储桶名称只能由小写字母、数字、句点（.）和连字符（-）组成。
- 存储桶名称必须以字母或数字开头和结尾。
- 存储桶名称不得包含两个相邻的句点。
- 存储桶名称不得采用 IP 地址格式（例如，192.168.5.4）。
- 存储桶名称不得以前缀 `xn--` 开头。
- 存储桶名称不得以前缀 `sthree-` 开头。
- 存储桶名称不得以前缀 `sthree-configurator` 开头。
- 存储桶名称不得以前缀 `amzn-s3-demo-` 开头。
- 存储桶名称不得以后缀 `-s3alias` 结尾。此后缀是为接入点别名预留的。有关更多信息，请参阅 为您的 S3 存储桶接入点使用存储桶式别名。
- 存储桶名称不得以后缀 `--ol-s3` 结尾。此后缀是为对象 Lambda 接入点别名预留的。有关更多信息，请参阅 如何为您的 S3 存储桶对象 Lambda 接入点使用存储桶式别名。
- 存储桶名称不得以后缀 `.mrap` 结尾。此后缀预留用于多区域接入点名称。有关更多信息，请参阅 命名 Amazon S3 多区域接入点的规则。

- 存储桶名称不得以后缀 `--x-s3` 结尾。此后缀预留用于目录存储桶。有关更多信息，请参阅 目录存储桶命名规则。
- 存储桶名称在分区内所有 AWS 区域中的所有 AWS 账户间必须是唯一的。分区是一组区域。AWS 目前有三个分区：`aws`（标准区域）、`aws-cn`（中国区域）和 `aws-us-gov`（AWS GovCloud (US)）。
- 存储桶名称不能被同一分区中的另一个 AWS 账户使用，直到存储桶被删除。
- 与 Amazon S3 Transfer Acceleration 一起使用的存储桶名称中不能有句点（.）。有关 Transfer Acceleration 的更多信息，请参阅使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输。

为了获得最佳兼容性，我们建议您避免在存储桶名称中使用句点（.），但仅用于静态网站托管的存储桶除外。如果您在存储桶名称中包含句点，则无法通过 HTTPS 使用虚拟主机式寻址，除非您执行自己的证书验证。这是因为用于存储桶虚拟托管的安全证书不适用于名称中带有句点的存储桶。

此限制不会影响用于静态网站托管的存储桶，因为静态网站托管只能通过 HTTP 提供。有关虚拟主机式寻址的更多信息，请参阅存储桶的虚拟托管。有关静态网站托管的更多信息，请参阅使用 Amazon S3 托管静态网站。

> **Note**
>
> 2018 年 3 月 1 日之前，在美国东部（弗吉尼亚北部）区域中创建的存储桶的名称最多可包含 255 个字符，并且包括大写字母和下划线。自 2018 年 3 月 1 日起，美国东部（弗吉尼亚北部）中的新存储桶必须符合在所有其他区域中应用的相同规则。

有关对象键名称的信息，请参阅创建对象键名称。

通用存储桶名称示例

以下示例存储桶名称是有效的，并遵循建议的通用存储桶命名准则：

- `docexamplebucket1`
- `log-delivery-march-2020`
- `my-hosted-content`

以下示例存储桶名称是有效的，但不推荐用于静态网站托管以外的其他用途：

- docexamplewebsite.com
- www.docexamplewebsite.com
- my.example.s3.bucket

以下示例存储桶名称无效：

- doc_example_bucket（包含下划线）
- DocExampleBucket（包含大写字母）
- doc-example-bucket-（以连字符结尾）

目录存储桶命名规则

以下命名规则适用于目录存储桶。

- 在选定的 AWS 区域和可用区内是唯一的。
- 名称的长度必须介于 3（最小）到 63（最大）个字符之间，包括后缀。
- 仅包含小写字母、数字和连字符（-）。
- 以字母或数字开头和结尾。
- 必须包含以下后缀：--*azid*--x-s3。
- 存储桶名称不得以前缀 xn-- 开头。
- 存储桶名称不得以前缀 sthree- 开头。
- 存储桶名称不得以前缀 sthree-configurator 开头。
- 存储桶名称不得以前缀 amzn-s3-demo- 开头。
- 存储桶名称不得以后缀 -s3alias 结尾。此后缀是为接入点别名预留的。有关更多信息，请参阅 为您的 S3 存储桶接入点使用存储桶式别名。
- 存储桶名称不得以后缀 --ol-s3 结尾。此后缀是为对象 Lambda 接入点别名预留的。有关更多信息，请参阅 如何为您的 S3 存储桶对象 Lambda 接入点使用存储桶式别名。
- 存储桶名称不得以后缀 .mrap 结尾。此后缀预留用于多区域接入点名称。有关更多信息，请参阅 命名 Amazon S3 多区域接入点的规则。

> **Note**
>
> 当您使用控制台创建目录存储桶时，系统会自动将后缀添加到您提供的基本名称。此后缀包括您选择的可用区的可用区 ID。
> 当您使用 API 创建目录存储桶时，必须在请求中提供包括可用区 ID 在内的完整后缀。有关可用区 ID 的列表，请参阅 S3 Express One Zone 可用区和区域。

访问和列出 Amazon S3 存储桶

您可以使用各种工具列出和访问 Amazon S3 存储桶。查看以下工具，确定哪种方法适合您的使用场景：

- Amazon S3 控制台：使用 Amazon S3 控制台，您可以轻松访问存储桶并修改存储桶属性。您还可以使用控制台 UI 执行大多数存储桶操作，而不必编写任何代码。
- AWS CLI：如果您需要访问多个存储桶，则可以使用 AWS Command Line Interface（AWS CLI）自动执行常见任务和重复任务，从而节省时间。随着组织的扩大，常见操作的可脚本化和可重复性是经常考虑的因素。有关更多信息，请参阅 使用 AWS CLI 进行 Amazon S3 开发。
- Amazon S3 REST API：您可以使用 Amazon S3 REST API 编写自己的程序并以编程方式访问存储桶。在 Amazon S3 支持的 API 架构中，存储桶和对象是资源，各自具有唯一标识资源的资源 URI。有关更多信息，请参阅 使用 REST API 进行 Amazon S3 开发。

根据您的 Amazon S3 存储桶的应用场景，建议使用不同方法来访问存储桶中的底层数据。以下列表包括访问数据的常见应用场景。

- 静态网站 – 您可以使用 Amazon S3 托管静态网站。在此应用场景中，您可以配置 S3 存储桶，使其功能与网站相似。要了解有关在 Amazon S3 上托管网站的步骤的示例，请参阅教程：在 Amazon S3 上配置静态网站。

  要托管启用了“屏蔽公共访问权限”等安全设置的静态网站，建议使用带来源访问控制（OAC）的 Amazon CloudFront，并实施其它安全标头，例如 HTTPS。有关更多信息，请参阅安全静态网站入门。

> **Note**
>
> Amazon S3 针对静态网站访问支持虚拟托管类型和路径类型 URL。由于可以使用路径类型和虚拟托管类型 URL 访问存储桶，因此，建议您使用符合 DNS 标准的存储桶名称创建存储桶。有关更多信息，请参阅 存储桶配额、限制和局限性。

- 共享数据集 – 在 Amazon S3 上扩展时，通常会采用多租户模式，即在共享存储桶中为不同的最终客户或业务部门分配唯一前缀。通过使用 Amazon S3 接入点，您可以将一个大型存储桶策略划分为每个需要访问共享数据集的应用程序的独立接入点策略。这种方法可以更轻松地专注于为一个应用程序构建正确的访问策略，而不中断任何其它应用程序在共享数据集中的操作。有关更多信息，请参阅使用 Amazon S3 接入点管理数据访问。
- 高吞吐量工作负载 – 适用于 Amazon S3 的 Mountpoint 是一款高吞吐量开源文件客户端，用于将 Amazon S3 存储桶作为本地文件系统进行挂载。借助 Mountpoint，您的应用程序可以通过文件系统操作（例如打开和读取）访问存储在 Amazon S3 中的对象。Mountpoint 会自动将这些操作转换为 S3 对象 API 调用，让您的应用程序能够通过文件接口访问 Amazon S3 的弹性存储和吞吐量。有关更多信息，请参阅 使用适用于 Amazon S3 的 Mountpoint。
- 多区域应用程序 – Amazon S3 多区域接入点提供了一个全局端点，应用程序可以使用该端点来满足来自位于多个 AWS 区域 中的 S3 存储桶的请求。您可以使用多区域接入点通过单个区域中使用的相同架构构建多区域应用程序，然后在世界任何地方运行这些应用程序。多区域接入点不是通过公共互联网发送请求，而是提供内置的网络韧性，加速向 Amazon S3 发送基于互联网的请求。有关更多信息，请参阅 Amazon S3 中的多区域接入点。
- 构建新应用程序 – 使用 Amazon S3 开发应用程序时，您可以使用 AWS SDK。AWS SDK 包装了底层 Amazon S3 REST API，可以简化您的编程任务。要构建互连移动应用程序和 Web 应用程序，您可以使用 AWS 移动 SDK 和 AWS Amplify JavaScript 库。有关更多信息，请参阅 使用 AWS SDK 通过 Amazon S3 进行开发。
- Secure Shell（SSH）文件传输协议（SFTP）– 如果您想通过互联网安全地传输敏感数据，则可以将支持 SFTP 的服务器与 Amazon S3 存储桶一起使用。AWS SFTP 是一种网络协议，支持 SSH 的全部安全和身份验证功能。使用此协议，您可以对用户身份、权限和密钥进行精细控制，也可以使用 IAM 策略来管理访问权限。要将支持 SFTP 的服务器与您的 Amazon S3 存储桶相关联，请务必先创建支持 SFTP 的服务器。然后设置用户账户，并将服务器与 Amazon S3 存储桶关联。有关此过程的介绍，请参阅 AWS 博客中的 AWS Transfer for SFTP – Fully Managed SFTP Service for Amazon S3。

列出存储桶

要列出您的所有存储桶，您必须拥有 `s3:ListAllMyBuckets` 权限。要访问存储桶，请确保您还获得了列出指定存储桶内容所必需的 AWS Identity and Access Management（IAM）权限。有关授予对 S3 存储桶的访问权限的示例存储桶策略，请参阅允许 IAM 用户访问某个存储桶。如果您遇到 HTTP 拒绝访问（403 禁止）错误，请参阅桶策略和 IAM policy。

您可以使用 Amazon S3 控制台、AWS CLI 或 AWS SDK 列出存储桶。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 从通用存储桶列表中，选择您想要查看的存储桶。

> **Note**
>
> 通用存储桶列表包括位于所有 AWS 区域中的存储桶。

使用 AWS CLI

要使用 AWS CLI 访问 S3 存储桶或生成 S3 存储桶列表，请使用 `ls` 命令。在列出存储桶中的所有对象时，请注意，您必须拥有 `s3:ListBucket` 权限。

要使用此示例命令，请将 *DOC-EXAMPLE-BUCKET1* 替换为您的存储桶的名称。

```
$ aws s3 ls s3://DOC-EXAMPLE-BUCKET1
```

以下示例命令将列出您账户中的所有 Amazon S3 存储桶：

```
$ aws s3 ls
```

有关更多信息和示例，请参阅列出存储桶和对象。

使用 AWS SDK

您也可以使用 ListBuckets API 操作访问 Amazon S3 存储桶。有关如何将此操作用于不同 AWS SDK 的示例，请参阅将 `ListBuckets` 与 AWS SDK 或 CLI 配合使用。

创建存储桶

要向 Amazon S3 上传数据，您必须首先在其中一个 AWS 区域 中创建 Amazon S3 存储桶。创建存储桶时，您必须选择存储桶名称和区域。您可以选择为存储桶选择其他存储管理选项。创建存储桶后，无法更改存储桶名称或区域。有关如何命名存储桶的信息，请参阅存储桶命名规则。

创建存储桶的 AWS 账户 拥有该存储桶。您可以将任何数量的对象上传到该存储桶。默认情况下，您可以在每个 AWS 账户中创建多达 100 个存储桶。如果您需要更多存储桶，则可以通过提交服务限制提高请求将账户的存储桶限制提高至最多 1,000 个存储桶。要了解如何提交存储桶限制提升，请参阅《AWS 一般参考》中的 AWS 服务限额。一个存储桶中可以存储任意数量的对象。

S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权，并禁用或启用访问控制列表（ACL）。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的每个对象，并使用策略专门管理对数据的访问权限。

有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。 。

具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）是 Amazon S3 中每个存储桶的基本加密配置级别。上传到 S3 存储桶的所有新对象都将使用 SSE-S3 作为基本加密级别设置自动进行加密。如果您想使用其他类型的默认加密，也可以指定具有 AWS Key Management Service（AWS KMS）密钥（SSE-KMS）或客户提供的密钥（SSE-C）的服务器端加密来加密数据。有关更多信息，请参阅 为 Amazon S3 存储桶设置默认服务器端加密行为。

您可以使用 Amazon S3 控制台、Amazon S3 API、AWS CLI 或 AWS SDK 创建存储桶。有关创建存储桶所需权限的更多信息，请参阅 Amazon Simple Storage Service API 参考中的 CreateBucket。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要在其中创建存储桶的区域。

> **Note**
>
> 要最大程度地减少延迟和成本以及满足法规要求，请选择一个靠近您的区域。在某一区域存储的对象将一直留在该区域，除非您特意将其转移到其他区域。有关 Amazon S3 AWS 区域的列表，请参阅《Amazon Web Services 一般参考》中的 AWS 服务 端点。

3. 在左侧导航窗格中，选择存储桶。
4. 选择创建存储桶。

   此时将打开创建存储桶页面。
5. 在常规配置下，查看将在其中创建存储桶的 AWS 区域。
6. 在存储桶类型下，选择通用。
7. 对于存储桶名称，请输入存储桶的名称。

   存储桶名称必须满足以下要求：

   - 在分区中是唯一的。分区是一组区域。AWS 目前有三个分区：aws（标准区域）、aws-cn（中国区域）和 aws-us-gov（AWS GovCloud (US) Regions）。
   - 长度必须介于 3 到 63 个字符之间。
   - 只能由小写字母、数字、句点（.）和连字符（-）组成。为了获得最佳兼容性，我们建议您避免在存储桶名称中使用句点（.），但仅用于静态网站托管的存储桶除外。
   - 以字母或数字开头和结尾。

   创建存储桶后，便无法再更改其名称。有关给存储桶命名的更多信息，请参阅存储桶命名规则。

   > **Important**
   >
   > 避免在存储桶名称中包含敏感信息，如账号。存储桶名称会显示在指向存储桶中的对象的 URL 中。

8. AWS Management Console可让您将现有存储桶的设置复制到新存储桶。如果您不想复制现有存储桶的设置，请跳到下一步。

   > **Note**
   >
   > 此选项：
   >
   > - 在 AWS CLI 中不可用，仅在控制台中可用
   > - 不可用于目录存储桶
   > - 不会将存储桶策略从现有存储桶复制到新存储桶

要复制现有存储桶的设置，请在从现有存储桶复制设置下，选择选择存储桶。将打开选择存储桶窗口。找到包含您要复制的设置的存储桶，然后选择选择存储桶。选择存储桶窗口关闭，创建存储桶窗口重新打开。

在从现有存储桶复制设置下，您现在将看到所选存储桶的名称。您还将看到还原默认值选项，您可以使用该选项移除复制的存储桶设置。在创建存储桶页面上查看其余存储桶设置。您将看到它们现在与您选择的存储桶的设置相匹配。您可以跳到最后一步。

9. 在 Object Ownership（对象所有权）下方，要禁用或启用 ACL，并控制上传到存储桶中的对象的所有权，请选择以下设置之一：

   已禁用 ACL

   - 强制存储桶拥有者（默认）– ACL 已禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响对 S3 存储桶中数据的访问权限。存储桶专门使用策略来定义访问控制。

     默认情况下，ACL 处于禁用状态。Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有必须单独控制每个对象的访问权限的特殊情况。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

   已启用 ACL

   - Bucket owner preferred（首选存储桶拥有者）— 存储桶拥有者拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 写入存储桶的新对象。

     如果应用首选存储桶拥有者设置，以要求所有 Amazon S3 上传都包含 `bucket-owner-full-control` 标准 ACL，则可以添加存储桶策略，该策略只允许使用此 ACL 上传对象。
   - 对象编写者— 该 AWS 账户上传对象拥有该对象，对其拥有完全控制权，并且可以通过 ACL 授予其他用户访问该对象的权限。

   > **Note**
   >
   > 默认设置为强制存储桶拥有者。要应用默认设置并将 ACL 保持为禁用状态，只需要 `s3:CreateBucket` 权限。要启用 ACL，您必须具有 `s3:PutBucketOwnershipControls` 权限。

10. 在此存储桶的屏蔽公共访问权限设置中，请选择要应用于存储桶的屏蔽公共访问权限设置。

默认情况下，启用所有四个“屏蔽公共访问权限”设置。我们建议您将所有设置保持为启用状态，除非您知道您需要为您的特定使用案例关闭其中一个或多个设置。有关屏蔽公共访问权限的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

> **Note**
>
> 要启用所有“屏蔽公共访问权限”设置，只需要 `s3:CreateBucket` 权限。要关闭任何“屏蔽公共访问权限”设置，您必须拥有 `s3:PutBucketPublicAccessBlock` 权限。

11. （可选）在 Bucket Versioning（存储桶版本控制）下，您可以选择是否要在存储桶中保留对象的变体。有关版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。

    要在存储桶上禁用或启用版本控制，请选择 Disable（禁用）或 Enable（启用）。

12. （可选）在 Tags（标签）下，您可以选择向存储桶添加标签。标签是用于对存储进行分类的键/值对。

    要添加存储桶标签，请输入 Key（键），并（可选）输入 Value（值），然后选择 Add Tag（添加标签）。

13. 在默认加密下，请选择编辑。

14. 要配置默认加密，请在加密类型下，选择以下选项之一：

    - Amazon S3 托管密钥（SSE-S3）
    - AWS Key Management Service 密钥（SSE-KMS）

    > **Important**
    >
    > 如果您将 SSE-KMS 选项用于默认加密配置，则您将受到 AWS KMS 的每秒请求数（RPS）限额限制。有关 AWS KMS 限额以及如何请求增加限额的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的限额。

    存储桶和新对象使用具有 Amazon S3 托管密钥的服务器端加密进行加密，作为加密配置的基本级别。有关默认加密的更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

    有关使用 Amazon S3 服务器端加密对数据进行加密的更多信息，请参阅使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

15. 如果选择了 AWS Key Management Service 密钥（SSE-KMS），则执行以下操作：

a. 在 AWS KMS 密钥下，通过以下方式之一指定您的 KMS 密钥：

- 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 密钥中进行选择，并从可用密钥的列表中选择您的 KMS 密钥。

  AWS 托管式密钥 (aws/s3) 和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。
- 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
- 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

  有关创建 AWS KMS key 的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

> **Important**
>
> 您只能使用与存储桶所在相同的 AWS 区域中可用的 KMS 密钥。Amazon S3 控制台仅列出与存储桶位于同一区域中的前 100 个 KMS 密钥。要使用未列出的 KMS 密钥，您必须输入 KMS 密钥 ARN。如果您希望使用其他账户拥有的 KMS 密钥，则必须首先有权使用该密钥，然后必须输入相应的 KMS 密钥 ARN。有关 KMS 密钥的跨账户权限的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建其他账户可以使用的 KMS 密钥。有关 SSE-KMS 的更多信息，请参阅使用 AWS KMS (SSE-KMS) 指定服务器端加密。
>
> 在 Amazon S3 中使用 AWS KMS key 进行服务器端加密时，您必须选择对称加密 KMS 密钥。Amazon S3 仅支持对称加密 KMS 密钥，而不支持非对称 KMS 密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的确定对称和非对称 KMS 密钥。

有关创建 AWS KMS key 的更多信息，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。有关将 AWS KMS 与 Amazon S3 结合使用的更多信息，请参阅使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。

b. 将存储桶配置为使用 SSE-KMS 进行默认加密时，您还可以启用 S3 存储桶密钥。S3 存储桶密钥可通过减少从 Amazon S3 到 AWS KMS 的请求流量，降低加密成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

要使用 S3 存储桶密钥，请在 Bucket Key（存储桶密钥）下，选择 Enable（启用）。

16. （可选）如果要启用 S3 对象锁定，请执行以下操作：

    a. 选择高级设置。

    > **Important**
    >
    > 启用对象锁定还会启用存储桶的版本控制。启用后，必须配置对象锁定默认保留和法律保留设置，以保护新对象不被删除或覆盖。

    b. 如果要启用对象锁定，请选择 Enable（启用），阅读出现的警告，并予以确认。

    有关更多信息，请参阅 使用 S3 对象锁定。

    > **Note**
    >
    > 要创建启用对象锁定的存储桶，您必须具有以下权限：`s3:CreateBucket`、`s3:PutBucketVersioning` 和 `s3:PutBucketObjectLockConfiguration`。

17. 请选择 Create bucket（创建存储桶）。

使用 AWS 软件开发工具包

使用 AWS SDK 创建存储桶时，您必须先创建一个客户端，然后使用该客户端发送创建存储桶的请求。作为最佳做法，您应在同一 AWS 区域 中创建客户端和存储桶。如果您在创建客户端或存储桶时未指定区域，Amazon S3 将使用默认区域，即美国东部（弗吉尼亚州北部）。如果您想将存储桶的创建限制在特定 AWS 区域内，则使用 `LocationConstraint` 条件键。

要创建客户端来访问双堆栈端点，则必须指定 AWS 区域。有关更多信息，请参阅 双堆栈端点。有关可用 AWS 区域的列表，请参阅《AWS 一般参考》中的区域和端点。

创建客户端时，区域映射到特定于区域的端点。客户端使用此端点与 Amazon S3 进行通信：`s3.region.amazonaws.com`。如果您的区域是在 2019 年 3 月 20 日之后启动的，则您的客户端和存储桶必须位于同一区域。不过，您可以使用美国东部（弗吉尼亚北部）区域中的客户端在 2019 年 3 月 20 日之前启动的任何区域中创建存储桶。有关更多信息，请参阅 传统终端节点。

这些 AWS SDK 代码示例执行以下任务：

- 通过显式指定 AWS 区域创建客户端 – 在本例中，客户端使用 `s3.us-west-2.amazonaws.com` 端点与 Amazon S3 通信。您可以指定任意 AWS 区域。有关 AWS 区域 的列表，请参阅《AWS 一般参考》中的区域和端点。
- 通过仅指定存储桶名称来发送创建存储桶请求 – 客户端向 Amazon S3 发送请求，请求在您创建客户端的区域中创建存储桶。
- 检索有关存储桶位置的信息 – Amazon S3 将存储桶位置信息存储在与存储桶关联的位置子资源中。

Java

此示例说明如何使用 AWS SDK for Java 创建 Amazon S3 存储桶。有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.GetBucketLocationRequest;

import java.io.IOException;

public class CreateBucket2 {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            if (!s3Client.doesBucketExistV2(bucketName)) {
                // Because the CreateBucketRequest object doesn't specify a region,
 the
                // bucket is created in the region specified in the client.
                s3Client.createBucket(new CreateBucketRequest(bucketName));
```

```
                // Verify that the bucket was created by retrieving it and checking
 its
                // location.
                String bucketLocation = s3Client.getBucketLocation(new
 GetBucketLocationRequest(bucketName));
                System.out.println("Bucket location: " + bucketLocation);
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

有关如何创建和测试有效示例的信息，请参阅 AWS SDK for .NET Version 3 API Reference。

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.S3.Util;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CreateBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
```

```
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CreateBucketAsync().Wait();
        }

        static async Task CreateBucketAsync()
        {
            try
            {
                if (!(await AmazonS3Util.DoesS3BucketExistAsync(s3Client,
bucketName)))
                {
                    var putBucketRequest = new PutBucketRequest
                    {
                        BucketName = bucketName,
                        UseClientRegion = true
                    };

                    PutBucketResponse putBucketResponse = await
s3Client.PutBucketAsync(putBucketRequest);
                }
                // Retrieve the bucket location.
                string bucketLocation = await FindBucketLocationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }
        static async Task<string> FindBucketLocationAsync(IAmazonS3 client)
        {
            string bucketLocation;
            var request = new GetBucketLocationRequest()
            {
                BucketName = bucketName
            };
            GetBucketLocationResponse response = await
client.GetBucketLocationAsync(request);
```

```
            bucketLocation = response.Location.ToString();
            return bucketLocation;
        }
    }
}
```

Ruby

有关如何创建和测试有效示例的信息，请参阅 AWS SDK for Ruby - Version 3。

Example

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
```

```
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 AWS CLI

您也可以使用AWS Command Line Interface（AWS CLI）创建 S3 存储桶。有关更多信息，请参阅《AWS CLI 命令参考》中的 create-bucket。

有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的什么是 AWS Command Line Interface？。

查看 S3 存储桶的属性

您可以查看您拥有的任何 Amazon S3 存储桶的属性。这些设置包括：

- Bucket Versioning（存储桶版本控制）– 使用版本控制在一个存储桶中保留对象的多个版本。默认情况下，将为新存储桶禁用版本控制。有关启用版本控制的信息，请参阅在存储桶上启用版本控制。
- 标签 – 利用 AWS 成本分配功能，您可以使用存储桶标签对存储桶的使用计费添加注释。一个标签即为一个键值对，用于表示用户分配给存储桶的标记。有关更多信息，请参阅 使用成本分配 S3 存储桶标签。
- Default encryption (默认加密) – 启用默认加密可为您提供自动服务器端加密。Amazon S3 会在将对象保存到磁盘之前对其进行加密，并在下载对象时对其进行解密。有关更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

- Server access logging（服务器访问日志记录）– 使用服务器访问日志记录详细地记录对您的存储桶提出的各种请求。默认情况下，Amazon S3 不会收集服务器访问日志。有关启用服务器访问日志记录的信息，请参阅启用 Amazon S3 服务器访问日志记录。
- AWS CloudTrail 数据事件 – 使用 CloudTrail 记录数据事件。默认情况下，跟踪不记录数据事件。记录数据事件将收取额外费用。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的记录数据事件以便跟踪。
- Event notifications（事件通知）– 启用特定的 Amazon S3 存储桶事件以在每次发生这些事件时向目标发送通知消息。有关更多信息，请参阅 使用 Amazon S3 控制台启用和配置事件通知。
- Transfer acceleration（传输加速）– 在您的客户端与 S3 存储桶之间实现快速、轻松和安全的远距离文件传输。有关启用传输加速的信息，请参阅启用和使用 S3 Transfer Acceleration。
- Object Lock (对象锁定) – 使用 S3 对象锁定在固定的时间段内或无限期地阻止删除或覆盖对象。有关更多信息，请参阅使用 S3 对象锁定。
- Requester Pays（申请方付款）– 如果您希望申请方（而不是存储桶拥有者）支付请求和数据传输费用，请启用申请方付款。有关更多信息，请参阅使用申请方付款存储桶进行存储传输和使用。
- Static website hosting (静态网站托管) – 您可以在 Amazon S3 上托管静态网站。有关更多信息，请参阅 使用 Amazon S3 托管静态网站。

您可以使用 AWS Management Console、AWS CLI 或 AWS SDK 查看存储桶属性

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，选择要查看其属性的存储桶的名称。
3. 选择属性选项卡。
4. 在属性页面上，您可以为存储桶配置上述属性。

使用 AWS CLI

使用 AWS CLI 查看存储桶属性

以下命令展示如何使用 AWS CLI 列出不同的存储桶属性。

以下命令返回与存储桶 *amzn-s3-demo-bucket1* 关联的标签集。有关存储桶标签的更多信息，请参阅使用成本分配 S3 存储桶标签。

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket1
```

有关更多信息和示例，请参阅《AWS CLI 命令参考》中的 get-bucket-tagging。

以下命令返回存储桶 *amzn-s3-demo-bucket1* 的版本控制状态。有关存储桶版本控制的信息，请参阅在 S3 存储桶中使用版本控制。

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket1
```

有关更多信息和示例，请参阅《AWS CLI 命令参考》中的 get-bucket-versioning。

以下命令返回存储桶 *amzn-s3-demo-bucket1* 的默认加密配置。默认情况下，所有存储桶都有默认加密配置，该配置使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。有关存储桶默认加密的信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
```

有关更多信息和示例，请参阅《AWS CLI 命令参考》中的 get-bucket-encryption。

以下命令返回存储桶 *amzn-s3-demo-bucket1* 的通知配置。有关存储桶事件通知的信息，请参阅 Amazon S3 事件通知。

```
aws s3api get-bucket-notification-configuration --bucket amzn-s3-demo-bucket1
```

有关更多信息和示例，请参阅《AWS CLI 命令参考》中的 get-bucket-notification-configuration。

以下命令返回存储桶 *amzn-s3-demo-bucket1* 的日志记录状态。有关存储桶日志记录的信息，请参阅使用服务器访问日志记录来记录请求。

```
aws s3api get-bucket-logging --bucket amzn-s3-demo-bucket1
```

有关更多信息和示例，请参阅 AWS CLI 命令参考中的 get-bucket-logging。

使用 AWS SDK

有关如何使用 AWS SDK 返回存储桶属性（例如版本控制、标签等）的示例，请参阅使用 AWS SDK 对 Amazon S3 执行的操作 。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

清空存储桶

您可以使用 Amazon S3 控制台、AWS SDK 或 AWS Command Line Interface（AWS CLI）清空存储桶的内容。清空存储桶时，您将删除所有对象，但会保留存储桶。清空存储桶的操作无法撤消。当正在执行清空存储桶操作时添加到存储桶的对象可能被删除。必须先删除存储桶中的所有对象（包括所有对象版本和删除标记），然后才能删除存储桶本身。

在清空已启用或暂停 S3 版本控制的存储桶时，存储桶中的所有对象的所有版本都将被删除。有关更多信息，请参阅 使用启用版本控制的存储桶中的对象。

您还可以指定存储桶的生命周期配置以使对象过期，以便 Amazon S3 能删除这些对象。有关更多信息，请参阅 在存储桶上设置生命周期配置。要清空大型存储桶，我们建议您使用 S3 生命周期配置规则。生命周期到期是一个异步过程，因此规则可能需要花数天时间运行，然后才能清空存储桶。在 Amazon S3 首次运行该规则后，所有符合过期条件的对象都将被标记为删除。您不再为那些标记为删除的对象付费。有关更多信息，请参阅如何使用生命周期配置规则清空 Amazon S3 存储桶？。

使用 S3 控制台

您可以使用 Amazon S3 控制台清空存储桶，这将删除存储桶中的所有对象，而不删除存储桶。

清空 S3 存储桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket name（存储桶名称）列表中，选择要清空的存储桶的名称旁边的选项，然后选择 Empty（清空）。
3. 在 Empty bucket（清空存储桶）页面上，通过在文本字段中输入存储桶名称来确认要清空存储桶，然后选择 Empty（清空）。
4. 在清空桶：状态页面上监控桶清空过程的进度。

使用 AWS CLI

仅在存储桶未启用存储桶版本控制时可使用 AWS CLI 清空存储桶。如果未启用版本控制，您可以将 `rm`（删除）AWS CLI命令与 `--recursive` 参数结合使用来清空存储桶（或删除部分带特定键名前缀的对象）。

以下 rm 命令将删除拥有键名前缀 doc 的对象，例如，doc/doc1 和 doc/doc2。

```
$ aws s3 rm s3://bucket-name/doc --recursive
```

使用以下命令删除所有对象，而无需指定前缀。

```
$ aws s3 rm s3://bucket-name --recursive
```

有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的将 AWS CLI 与高级别 S3 命令结合使用。

> **Note**
>
> 您无法从启用了版本控制的存储桶中删除对象。当您删除一个对象时（此命令将执行的操作），Amazon S3 将添加一个删除标记。有关 S3 存储桶版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。

使用 AWS 开发工具包

您可以使用 AWS SDK 清空存储桶或删除部分拥有特定键名前缀的对象。

有关如何使用AWS SDK for Java清空存储桶的示例，请参阅删除存储桶。该代码将删除所有对象（不论存储桶是否启用了版本控制），然后删除存储桶。要只清空存储桶，请确保删除用来删除存储桶的语句。

有关使用其他 AWS SDK 的更多信息，请参阅用于 Amazon Web Services 的工具。

使用生命周期配置

要清空大型存储桶，我们建议您使用 S3 生命周期配置规则。生命周期到期是一个异步过程，因此规则可能需要花数天时间运行，然后才能清空存储桶。在 Amazon S3 首次运行该规则后，所有符合过期条件的对象都将被标记为删除。您不再为那些标记为删除的对象付费。有关更多信息，请参阅如何使用生命周期配置规则清空 Amazon S3 存储桶？。

如果使用生命周期配置清空存储桶，则配置应包括当前版本、非当前版本、删除标记和未完成的分段上传。

您可以添加生命周期配置规则，以使所有对象或部分拥有特定键名前缀的对象过期。例如，要删除存储桶中的所有对象，您可以将生命周期规则设置为使对象在创建一天后过期。

Amazon S3 支持一个存储桶生命周期规则，您可以使用该规则停止未在启动后的指定天数内完成的分段上传。我们建议您配置此生命周期规则以最大限度地降低存储成本。有关更多信息，请参阅 配置存储桶生命周期配置以删除未完成的分段上传。

有关使用生命周期配置清空存储桶的更多信息，请参阅在存储桶上设置生命周期配置和即将过期的对象。

清空已配置 AWS CloudTrail 的存储桶

AWS CloudTrail 跟踪 Amazon S3 存储桶中的对象级数据事件，例如删除对象。如果您使用存储桶作为记录 CloudTrail 事件的目标并从同一个存储桶中删除对象，则在清空存储桶的同时可能正在创建新对象。为了防止这种情况，请停止 AWS CloudTrail 跟踪。有关让 CloudTrail 跟踪停止记录事件的更多信息，请参阅《AWS CloudTrail 用户指南》中的关闭跟踪的日志记录。

停止将 CloudTrail 跟踪添加到存储桶中的另一种备选方法是：将 `s3:PutObject` 语句添加到存储桶策略中。如果您想稍后在存储桶中存储新对象，则需要删除此拒绝 `s3:PutObject` 语句。有关更多信息，请参阅《IAM 用户指南》中的 对象操作 和 IAM JSON 策略元素：效果。

删除存储桶

您可以删除空的 Amazon S3 存储桶。在删除存储桶之前，请注意以下事项：

- 存储桶名称是唯一的。如果删除存储桶，则另一个 AWS 用户可以使用该名称。
- 如果存储桶托管静态网站，并且您如 教程：使用注册到 Route 53 的自定义域配置静态网站 中所述创建并配置了 Amazon Route 53 托管区域，则必须清理与该存储桶相关的 Route 53 托管区域设置。有关更多信息，请参阅 步骤 2：删除 Route 53 托管区域。
- 如果该存储桶收到来自 Elastic Load Balancing（ELB）的日志数据：建议先停止将 ELB 日志传输到该存储桶，然后再删除该存储桶。删除该存储桶后，如果其他用户创建使用相同名称的存储桶，则日志数据可能会传输到此同名存储桶。有关 ELB 访问日志的信息，请参阅《经典负载均衡器用户指南》中的访问日志和《应用程序负载均衡器用户指南》中的访问日志。

故障排除

如果您无法删除 Amazon S3 存储桶，请考虑以下事项：

- Make sure the bucket is empty（确保存储桶为空）— 您只能删除其中没有任何对象的存储桶。确保存储桶为空。

- 确保没有附加任何接入点 — 您只能删除其中没有任何接入点的存储桶。在删除存储桶之前，请删除附加到该存储桶的所有接入点。
- AWS Organizations 服务控制策略（SCP）– 服务控制策略可以拒绝对存储桶的删除权限。有关 SCP 的信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- s3:DeleteBucket 权限 – 如果您无法删除存储桶，请与 IAM 管理员合作，确认您具有 `s3:DeleteBucket` 权限。有关如何查看或更新 IAM 权限的信息，请参阅《IAM 用户指南》中的更改 IAM 用户的权限。
- s3:DeleteBucket 拒绝语句 — 如果您在 IAM 策略中拥有 `s3:DeleteBucket` 权限且无法删除存储桶，则存储桶策略可能包含 `s3:DeleteBucket` 的拒绝语句。默认情况下，由 ElasticBeanstalk 创建的存储桶具有包含此语句的策略。您必须先删除此语句或存储桶策略，然后才能删除存储桶。

> **Important**
>
> 存储桶名称是唯一的。如果删除存储桶，则另一个 AWS 用户可以使用该名称。如果您希望继续使用相同的存储桶名称，请不要删除该存储桶。我们建议您，清空并保留存储桶。

使用 S3 控制台

删除 S3 存储桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要删除的存储桶名称旁边的选项，然后选择页面顶部的 Delete（删除）。
3. 在 Delete bucket（删除存储桶）页面上，通过在文本字段中输入存储桶名称来确认要删除存储桶，然后选择 Delete bucket（删除存储桶）。

> **Note**
>
> 如果存储桶包含任何对象，请在删除存储桶之前清空存储桶，具体操作如下：在 This bucket is not empty（此存储桶不为空）错误提醒中选择 empty bucket configuration（清空存储桶配置）链接，然后按照 Empty bucket（清空存储桶）页面上的说明操作。然后，返回到 Delete bucket（删除存储桶）页面并删除存储桶。

4. 要验证您是否已删除存储桶，请打开 Buckets（存储桶）列表并输入您删除的存储桶的名称。如果无法找到此存储桶，则表示删除成功。

使用适用于 Java 的 AWS SDK

以下示例展示了如何使用适用于 Java 的 AWS SDK 删除存储桶。首先，该代码将删除存储桶中的对象，然后删除存储桶。有关其他 AWS SDK 的信息，请参阅用于 Amazon Web Services 的工具。

Java

以下 Java 示例删除包含对象的存储桶。该示例将删除所有对象，然后删除存储桶。该示例适用于已启用版本控制或未启用版本控制的存储桶。

> **Note**
>
> 对于未启用版本控制的存储桶，您可以直接删除所有对象，然后删除存储桶。对于启用了版本控制的存储桶，您必须先删除所有对象版本，然后再删除存储桶。

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.Iterator;

public class DeleteBucket2 {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
```

```
                    .withRegion(clientRegion)
                    .build();

            // Delete all objects from the bucket. This is sufficient
            // for unversioned buckets. For versioned buckets, when you attempt to
delete
            // objects, Amazon S3 inserts
            // delete markers for all objects, but doesn't delete the object
versions.
            // To delete objects from versioned buckets, delete all of the object
versions
            // before deleting
            // the bucket (see below for an example).
            ObjectListing objectListing = s3Client.listObjects(bucketName);
            while (true) {
                Iterator<S3ObjectSummary> objIter =
objectListing.getObjectSummaries().iterator();
                while (objIter.hasNext()) {
                    s3Client.deleteObject(bucketName, objIter.next().getKey());
                }

                // If the bucket contains many objects, the listObjects() call
                // might not return all of the objects in the first listing. Check
to
                // see whether the listing was truncated. If so, retrieve the next
page of
                // objects
                // and delete them.
                if (objectListing.isTruncated()) {
                    objectListing = s3Client.listNextBatchOfObjects(objectListing);
                } else {
                    break;
                }
            }

            // Delete all object versions (required for versioned buckets).
            VersionListing versionList = s3Client.listVersions(new
ListVersionsRequest().withBucketName(bucketName));
            while (true) {
                Iterator<S3VersionSummary> versionIter =
versionList.getVersionSummaries().iterator();
                while (versionIter.hasNext()) {
                    S3VersionSummary vs = versionIter.next();
```

```
                    s3Client.deleteVersion(bucketName, vs.getKey(),
 vs.getVersionId());
                }

                if (versionList.isTruncated()) {
                    versionList = s3Client.listNextBatchOfVersions(versionList);
                } else {
                    break;
                }
            }

            // After all objects and object versions are deleted, delete the bucket.
            s3Client.deleteBucket(bucketName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
 couldn't
            // parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 AWS CLI

如果存储桶没有启用版本控制，则可以使用 AWS CLI 删除包含对象的存储桶。当您删除一个包含对象的存储桶时，该存储桶中的所有对象都将被永久删除，包括已转换为 S3 Glacier 存储类的对象。

如果存储桶未启用版本控制，则可将 rb（删除存储桶）AWS CLI命令和 --force 参数结合使用来删除存储桶及其中的所有对象。此命令将先删除所有对象，然后再删除存储桶。

如果启用了版本控制，则不会在此过程中删除版本控制对象，这将导致存储桶删除失败，因为存储桶不能为空。有关删除受版本控制的对象的更多信息，请参阅删除对象版本。

```
$ aws s3 rb s3://bucket-name --force
```

有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的将 AWS Command Line Interface 与高级别 S3 命令结合使用。

为 Amazon S3 存储桶设置默认服务器端加密行为

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

默认情况下，所有 Amazon S3 存储桶都配置了加密，并且通过具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）来自动加密对象。此加密设置适用于 Amazon S3 存储桶中的所有对象。

如果您需要对密钥进行更多控制，例如管理密钥轮换和访问策略授予，则可以选择使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS），或具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）。有关编辑 KMS 密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的编辑密钥。

> **Note**
>
> 我们已更改存储桶，以自动加密新上传的对象。如果您之前创建的存储桶没有默认加密，则 Amazon S3 在默认情况下将使用 SSE-S3 为存储桶启用加密。对于已配置 SSE-S3 或 SSE-KMS 的现有存储桶的默认加密配置，不会进行任何更改。如果您想使用 SSE-KMS 加密您的对象，则必须在存储桶设置中更改加密类型。有关更多信息，请参阅 使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。

当您将存储桶配置为使用 SSE-KMS 的默认加密时，您还可以启用 S3 存储桶密钥，以减少从 Amazon S3 到 AWS KMS 的请求流量并降低加密成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

要识别已启用 SSE-KMS 进行默认加密的存储桶，您可以使用 Amazon S3 Storage Lens 存储统计管理工具指标。S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围

内了解对象存储的使用情况和活动。有关更多信息，请参阅使用 S3 Storage Lens 存储统计管理工具保护您的数据。

在使用服务器端加密时，Amazon S3 在将对象保存到其磁盘上之前对其进行加密，并在下载对象时对其进行解密。有关使用服务器端加密和加密密钥管理来保护数据的更多信息，请参阅使用服务器端加密保护数据。

有关默认加密所需的权限的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketEncryption。

您可以使用 Amazon S3 控制台、AWS SDK、Amazon S3 REST API 和 AWS 命令行界面（AWS CLI）为 S3 存储桶配置 Amazon S3 默认加密行为。

加密现有对象

要加密现有未加密的 Amazon S3 对象，可以使用 Amazon S3 批量操作。您为 S3 批量操作提供了要操作的对象列表，而批量操作调用相应的 API 来执行指定的操作。您可以使用批量操作复制操作复制现有的未加密对象，并将其作为加密对象写回同一存储桶。单个批量操作作业可对数十亿个对象执行指定操作。有关更多信息，请参阅 对 Amazon S3 对象执行大规模批量操作 和 AWS 存储博客的博客文章：使用 Amazon S3 批量操作加密对象。

也可以使用 `CopyObject` API 操作或 `copy-object` AWS CLI 命令加密现有对象。有关更多信息，请参阅 AWS 存储博客的博客文章：使用 AWS CLI 加密现有 Amazon S3 对象。

> **Note**
>
> 将默认存储桶加密设置为 SSE-KMS 的 Amazon S3 存储桶不能用作 the section called “记录服务器访问” 的目标存储桶。对于服务器访问日志目标存储桶，仅支持 SSE-S3 默认加密。

使用 SSE-KMS 加密进行跨账户操作

在对跨账户操作使用加密时，请注意以下事项：

- 如果未在请求时提供 AWS KMS key Amazon 资源名称（ARN）或别名，也未通过存储桶的默认加密配置提供它们时，则使用 AWS 托管式密钥（`aws/s3`）。
- 如果您使用您的 KMS 密钥所在的相同 AWS 账户中的 AWS Identity and Access Management（IAM）主体上传或访问 S3 对象，则可以使用 AWS 托管式密钥（`aws/s3`）。
- 如果您希望授予对 S3 对象的跨账户访问权限，请使用客户自主管理型密钥。您可以配置客户托管式密钥的策略，以便允许从其他账户进行访问。

- 如果您指定客户托管式 KMS 密钥，我们建议您使用完全限定的 KMS 密钥 ARN。如果您改为使用 KMS 密钥别名，AWS KMS 将解析请求者账户中的密钥。这一行为可能导致使用属于请求者而不是存储桶拥有者的 KMS 密钥来加密数据。
- 您必须指定您（请求者）已被授予 `Encrypt` 权限的密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的允许密钥用户使用 KMS 密钥进行加密操作。

有关何时使用客户自主管理型密钥和 AWS KMS 托管密钥的更多信息，请参阅我是应使用 AWS 托管式密钥 还是使用客户自主管理型密钥来加密 Amazon S3 中的对象？

将默认加密用于复制

在为复制目标存储桶启用默认加密后，将应用以下加密行为：

- 如果未对源存储桶中的对象进行加密，则将使用目标存储桶的默认加密设置对目标存储桶中的副本对象进行加密。因此，源对象的实体标签（ETag）与副本对象的 ETag 不同。如果您有使用 ETag 的应用程序，则必须更新这些应用程序以弥补这种差异。
- 如果使用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）、具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）或具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）来加密源存储桶中的对象，则目标存储桶中的副本对象使用与源对象相同类型的加密。不会使用目标存储桶的默认加密设置。

有关使用 SSE-KMS 进行默认加密的更多信息，请参阅复制加密对象。

将 Amazon S3 存储桶密钥用于默认加密

当将存储桶配置为使用 SSE-KMS 作为新对象的默认加密行为时，还可以配置 S3 存储桶密钥。S3 存储桶密钥可减少从 Amazon S3 到 AWS KMS 的事务数量，从而降低 SSE-KMS 的成本。

当您将存储桶配置为使用 S3 存储桶密钥对新对象进行 SSE-KMS 加密时，AWS KMS 会生成存储桶级别密钥，该密钥用于为存储桶中的对象创建唯一的数据密钥。此 S3 存储桶密钥在 Amazon S3 内限时使用，从而减少了 Amazon S3 向 AWS KMS 发出请求，以完成加密操作的需求。

有关使用 S3 存储桶密钥的更多信息，请参阅 使用 Amazon S3 存储桶密钥。

配置默认加密

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

默认设置情况下，Amazon S3 存储桶启用了存储桶加密，并且通过具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）来自动加密新对象。这种加密适用于您的 Amazon S3 存储桶中的所有新对象，并且不收取任何费用。

如果您需要对加密密钥进行更多控制，例如管理密钥轮换和访问策略授予，则可以选择使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS），或具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）。有关 SSE-KMS 的更多信息，请参阅 使用 AWS KMS (SSE-KMS) 指定服务器端加密。有关 DSSE-KMS 的更多信息，请参阅the section called “双层服务器端加密（DSSE-KMS）”。

如果您希望使用其他账户拥有的 KMS 密钥，则您必须有权使用该密钥。有关 KMS 密钥的跨账户权限的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建其他账户可以使用的 KMS 密钥。

当您将默认存储桶加密设置为 SSE-KMS 时，还可以配置 S3 存储桶密钥以降低 AWS KMS 请求成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

> **Note**
>
> 如果使用 PutBucketEncryption 将默认存储桶加密设置为 SSE-KMS，则应验证您的 KMS 密钥 ID 是否正确。Amazon S3 不验证 PutBucketEncryption 请求中提供的 KMS 密钥 ID。

对 S3 存储桶使用默认加密不会产生额外的费用。请求配置默认加密行为会产生标准 Amazon S3 请求费用。有关定价的信息，请参阅 Amazon S3 定价。对于 SSE-KMS 和 DSSE-KMS，将会产生 AWS KMS 费用，这些费用在 AWS KMS 定价中列出。

默认加密不支持具有客户提供的密钥的服务器端加密（SSE-C）。

您可以使用 Amazon S3 控制台、AWS SDK、Amazon S3 REST API 和 AWS Command Line Interface（AWS CLI）为 S3 存储桶配置 Amazon S3 默认加密。

启用默认加密之前需要注意的更改

在为存储桶启用默认加密后，将会应用以下加密行为：

- 在启用默认加密之前，存储桶中已存在的对象的加密没有变化。
- 在启用默认加密后上传对象时：
  - 如果您的 PUT 请求标头不包含加密信息，则 Amazon S3 将使用存储桶的默认加密设置来加密对象。
  - 如果您的 PUT 请求标头包含加密信息，则 Amazon S3 将使用 PUT 请求中的加密信息加密对象，然后再将对象存储在 Amazon S3 中。
- 如果您将 SSE-KMS 或 DSSE-KMS 选项用于默认加密配置，则您将受到 AWS KMS 的每秒请求数（RPS）限额限制。有关 AWS KMS 限额以及如何请求增加限额的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的限额。

> **Note**
>
> 在启用默认加密之前上传的对象将不会加密。有关加密现有对象的信息，请参阅the section called “设置默认存储桶加密”。

使用 S3 控制台

在 Amazon S3 存储桶上配置默认加密

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（存储桶）列表中，请选择您想要的存储桶的名称。
4. 选择属性选项卡。
5. 在默认加密下，选择编辑。
6. 要配置加密，请在加密类型下，选择以下选项之一：

- 具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）
- 具有 AWS Key Management Service 密钥的服务器端加密（SSE-KMS）
- 具有 AWS Key Management Service 密钥的双层服务器端加密（DSSE-KMS）

> **Important**
>
> 如果您将 SSE-KMS 或 DSSE-KMS 选项用于默认加密配置，则您将受到 AWS KMS 的每秒请求数（RPS）限额限制。有关 AWS KMS 限额以及如何请求增加限额的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的限额。

默认情况下，存储桶和新对象使用 SSE-S3 加密，除非您为存储桶指定其他类型的默认加密。有关默认加密的更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

有关使用 Amazon S3 服务器端加密对数据进行加密的更多信息，请参阅使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

7. 如果您选择具有 AWS Key Management Service 密钥的服务器端加密（SSE-KMS）或具有 AWS Key Management Service 密钥的双层服务器端加密（DSSE-KMS），请执行以下操作：

   a. 在 AWS KMS 密钥下，通过以下方式之一指定您的 KMS 密钥：

      - 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 密钥中进行选择，并从可用密钥的列表中选择您的 KMS 密钥。

        AWS 托管式密钥 (aws/s3) 和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。
      - 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
      - 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

        有关创建 AWS KMS key 的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

> **Important**
>
> 您只能使用与存储桶所在相同 AWS 区域 中启用的 KMS 密钥。选择从您的 KMS 密钥中选择时，S3 控制台每个区域仅列出 100 个 KMS 密钥。如果在同一区域中有超过 100 个 KMS，您只会在 S3 控制台中看到前 100 个 KMS 密钥。要使用控制台中未列出的 KMS 密钥，请选择输入 AWS KMS key ARN，然后输入 KMS 密钥 ARN。
> 在 Amazon S3 中使用 AWS KMS key 进行服务器端加密时，您必须选择对称加密 KMS 密钥。Amazon S3 仅支持对称加密 KMS 密钥。有关这些密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的对称加密 KMS 密钥。

有关将 SSE-KMS 与 Amazon S3 结合使用的更多信息，请参阅使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。有关使用 DSSE-KMS 的更多信息，请参阅the section called “双层服务器端加密（DSSE-KMS）”。

b. 将存储桶配置为使用 SSE-KMS 进行默认加密时，您还可以启用 S3 存储桶密钥。S3 存储桶密钥可通过减少从 Amazon S3 到 AWS KMS 的请求流量，降低加密成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

   要使用 S3 存储桶密钥，请在 Bucket Key（存储桶密钥）下，选择 Enable（启用）。

> **Note**
>
> DSSE-KMS 不支持 S3 存储桶密钥。

8. 选择 Save changes（保存更改）。

使用 AWS CLI

这些示例说明如何使用 SSE-S3 或将 SSE-KMS 与 S3 存储桶密钥结合使用来配置默认加密。

有关默认加密的更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。有关使用 AWS CLI 配置默认加密的更多信息，请参阅 put-bucket-encryption。

Example – 使用 SSE-S3 进行默认加密

此示例使用 Amazon S3 托管密钥来配置默认存储桶加密。

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
    "Rules": [
        {
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }
    ]
}'
```

Example – 使用 S3 存储桶密钥通过 SSE-KMS 进行默认加密

此示例使用 S3 存储桶密钥通过 SSE-KMS 配置默认存储桶加密。

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
    "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

使用 REST API

使用 REST API `PutBucketEncryption` 操作可启用默认加密，并设置要使用的服务器端加密类型：SSE-S3、SSE-KMS 或 DSSE-KMS。

有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketEncryption。

使用 AWS CloudTrail 和 Amazon EventBridge 监控默认加密

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象

都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

您可以使用 AWS CloudTrail 事件跟踪 Amazon S3 存储桶的默认加密配置请求。CloudTrail 日志中使用以下 API 事件名称：

- `PutBucketEncryption`
- `GetBucketEncryption`
- `DeleteBucketEncryption`

您还可以创建 EventBridge 规则，以匹配这些 API 调用的 CloudTrail 事件。有关 CloudTrail 事件的更多信息，请参阅使用控制台为存储桶中的对象启用日志记录功能。有关 EventBridge 事件的更多信息，请参阅 AWS 服务中的事件。

您可以使用 CloudTrail 日志执行对象级 Amazon S3 操作，以跟踪向 Amazon S3 发出的 PUT 和 POST 请求。您可以使用这些操作来验证在传入 PUT 请求不包含加密标头时是否使用默认加密来加密对象。

当 Amazon S3 使用默认加密设置来加密对象时，日志将包含以下字段之一作为名称/值对：`"SSEApplied":"Default_SSE_S3"`、`"SSEApplied":"Default_SSE_KMS"` 或 `"SSEApplied":"Default_DSSE_KMS"`。

当 Amazon S3 使用 PUT 加密标头来加密对象时，日志将包含以下字段之一作为名称/值对：`"SSEApplied":"SSE_S3"`、`"SSEApplied":"SSE_KMS"`、`"SSEApplied":"DSSE_KMS"` 或 `"SSEApplied":"SSE_C"`。

对于分段上传，该信息包含在 `InitiateMultipartUpload` API 操作请求中。有关使用 CloudTrail 和 CloudWatch 的更多信息，请参阅监控 Amazon S3。

使用适用于 Amazon S3 的 Mountpoint

适用于 Amazon S3 的 Mountpoint 是一款高吞吐量的开源文件客户端，用于将 Amazon S3 存储桶作为本地文件系统进行挂载。借助 Mountpoint，您的应用程序可以通过文件系统操作（例如打开和读取）访问存储在 Amazon S3 中的对象。Mountpoint 会自动将这些操作转换为 S3 对象 API 调用，让您的应用程序能够通过文件接口访问 Amazon S3 的弹性存储和吞吐量。

适用于 Amazon S3 的 Mountpoint 通常可用于读取密集的大规模应用程序的生产用途：数据湖、机器学习训练、图像渲染、自动驾驶汽车模拟、提取、转换、加载 (ETL) 等。

Mountpoint 支持基本的文件系统操作，并且可以读取最大 5 TB 的文件。它可以列出和读取现有文件，也可以创建新文件。它无法修改现有文件或删除目录，也不支持符号链接或文件锁定。Mountpoint 非常适合具有下列特征的应用程序：不需要共享文件系统的全部功能和 POSIX 风格的权限，但需要 Amazon S3 的弹性吞吐量来读写大型 S3 数据集。有关更多信息，请参阅 GitHub 上的 Mountpoint file system behavior。对于需要完整 POSIX 支持的工作负载，我们建议使用适用于 Lustre 的 Amazon FSx 及其对链接 S3 存储桶的支持。

适用于 Amazon S3 的 Mountpoint 仅适用于 Linux 操作系统。您可以使用 Mountpoint 访问所有存储类中的 S3 对象，但 S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive、S3 Intelligent-Tiering Archive Access Tier 和 S3 Intelligent-Tiering Deep Archive Access Tier 除外。

主题

安装 Mountpoint

您可以使用命令行下载和安装适用于 Amazon S3 的 Mountpoint 的预构建程序包。根据您正在使用哪个 Linux 操作系统，下载和安装 Mountpoint 的说明会有所不同。

主题

基于 RPM 的发行版（Amazon Linux、Fedora、CentOS、RHEL）

1. 复制您的架构的以下下载网址。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm
   ```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm
```

2. 下载适用于 Amazon S3 的 Mountpoint 程序包。将 *download-link* 替换为上一步中的相应下载 URL。

```
wget download-link
```

3. （可选）验证已下载文件的真实性和完整性。首先，复制适用于您的架构的签名 URL。

x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm.asc
```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm.asc
```

接下来，请参阅验证适用于 Amazon S3 的 Mountpoint 程序包的签名。

4. 使用以下命令安装此程序包：

```
sudo yum install ./mount-s3.rpm
```

5. 输入以下命令，验证 Mountpoint 是否已成功安装：

```
mount-s3 --version
```

您应该可以看到类似于如下所示的输出内容：

```
mount-s3 1.3.1
```

基于 DEB 的发行版（Debian、Ubuntu）

1. 复制您的架构的下载网址。

x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb
```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb
```

2. 下载适用于 Amazon S3 的 Mountpoint 程序包。将 *download-link* 替换为上一步中的相应下载 URL。

```
wget download-link
```

3. （可选）验证已下载文件的真实性和完整性。首先，复制您的架构的签名 URL。

x86_64:

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb.asc
```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb.asc
```

接下来，请参阅验证适用于 Amazon S3 的 Mountpoint 程序包的签名。

4. 使用以下命令安装此程序包：

```
sudo apt-get install ./mount-s3.deb
```

5. 运行以下命令，验证适用于 Amazon S3 的 Mountpoint 是否已成功安装：

```
mount-s3 --version
```

您应该可以看到类似于如下所示的输出内容：

```
mount-s3 1.3.1
```

其他 Linux 发行版

1. 请查阅您的操作系统文档以安装必需的 FUSE 和 `libfuse2` 程序包。

2. 复制您的架构的下载网址。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz
   ```

3. 下载适用于 Amazon S3 的 Mountpoint 程序包。将 *download-link* 替换为上一步中的相应下载 URL。

   ```
   wget download-link
   ```

4. （可选）验证已下载文件的真实性和完整性。首先，复制您的架构的签名 URL。

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz.asc
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz.asc
   ```

   接下来，请参阅验证适用于 Amazon S3 的 Mountpoint 程序包的签名。

5. 使用以下命令安装此程序包：

   ```
   sudo mkdir -p /opt/aws/mountpoint-s3 && sudo tar -C /opt/aws/mountpoint-s3 -xzf ./
   mount-s3.tar.gz
   ```

6. 将 `mount-s3` 二进制文件添加到您的 `PATH` 环境变量。在您的 `$HOME/.profile` 文件中，附加下面一行：

   ```
   export PATH=$PATH:/opt/aws/mountpoint-s3/bin
   ```

   保存 `.profile` 文件，并运行以下命令：

```
source $HOME/.profile
```

7. 运行以下命令，验证适用于 Amazon S3 的 Mountpoint 是否已成功安装：

```
mount-s3 --version
```

您应该可以看到类似于如下所示的输出内容：

```
mount-s3 1.3.1
```

验证适用于 Amazon S3 的 Mountpoint 程序包的签名

1. 安装 GnuPG（gpg 命令）。需要验证所下载的适用于 Amazon S3 的 Mountpoint 程序包的真实性和完整性。默认情况下，GnuPG 安装在 Amazon Linux 亚马逊机器映像（AMI）上。安装 GnuPG 之后，继续执行步骤 2。
2. 通过运行以下命令下载 Mountpoint 公有密钥：

```
wget https://s3.amazonaws.com/mountpoint-s3-release/public_keys/KEYS
```

3. 通过运行以下命令将 Mountpoint 公有密钥导入到您的密钥环中：

```
gpg --import KEYS
```

4. 通过运行以下命令验证 Mountpoint 公有密钥的指纹：

```
gpg --fingerprint mountpoint-s3@amazon.com
```

确认显示的指纹字符串与以下内容相匹配：

```
673F E406 1506 BB46 9A0E  F857 BE39 7A52 B086 DA5A
```

如果指纹字符串不匹配，请不要完成安装 Mountpoint 并联系 AWS Support。

5. 下载软件包签名文件。将 *signature-link* 替换为前面各节中的相应签名链接。

```
wget signature-link
```

6. 通过运行以下命令验证下载的程序包的签名。将 *signature-filename* 替换为上一步中的文件名。

```
gpg --verify signature-filename
```

例如，在基于 RPM 的发行版（包括 Amazon Linux）上，运行以下命令：

```
gpg --verify mount-s3.rpm.asc
```

7. 输出应包含短语 `Good signature`。如果输出包含短语 `BAD signature`，请重新下载 Mountpoint 程序包文件并重复这些步骤。如果问题仍然存在，请不要完成 Mountpoint 的安装并联系 AWS Support。

输出可能包括有关可信签名的警告。这并不表示存在问题。这仅表示您尚未独立验证 Mountpoint 公有密钥。

配置和使用 Mountpoint

要使用适用于 Amazon S3 的 Mountpoint，您的主机需要有效的 AWS 凭证，有权访问您要挂载的一个或多个存储桶。有关不同的身份验证方式，请参阅 GitHub 上的 Mountpoint AWS 凭证。

例如，您可以创建一个新的 AWS Identity and Access Management（IAM）用户和角色用于此目的。请确保此角色有权访问您要挂载的一个或多个存储桶。您可以通过实例配置文件将 IAM 角色传递给您的 Amazon EC2 实例。

使用适用于 Amazon S3 的 Mountpoint

使用适用于 Amazon S3 的 Mountpoint 执行以下操作：

1. 使用 `mount-s3` 命令挂载存储桶。

在下面的示例中，将 *DOC-EXAMPLE-BUCKET* 替换为 S3 存储桶的名称，并将 *~/mnt* 替换为主机上您希望在其中挂载 S3 存储桶的目录。

```
mkdir ~/mnt
mount-s3 DOC-EXAMPLE-BUCKET ~/mnt
```

由于 Mountpoint 客户端在默认情况下在后台运行，因此 *~/mnt* 目录现在允许您访问 S3 存储桶中的对象。

2. 通过 Mountpoint 访问存储桶中的对象。

   在本地挂载存储桶后，您可以使用常用的 Linux 命令（例如 `cat` 或 `ls`）来处理您的 S3 对象。适用于 Amazon S3 的 Mountpoint 通过以正斜杠（`/`）字符为界拆分 S3 存储桶中的密钥，将这些密钥解释为文件系统路径。例如，如果您的存储桶中有对象密钥 `Data/2023-01-01.csv`，那么您的 Mountpoint 文件系统中将有一个名为 `Data` 的目录，其中有一个名称为 `2023-01-01.csv` 的文件。

   适用于 Amazon S3 的 Mountpoint 特意不实现文件系统的完整 POSIX 标准规范。Mountpoint 针对以下工作负载进行了优化：此类工作负载需要通过文件系统接口对存储在 Amazon S3 中的数据进行高吞吐量读写访问，但不依赖文件系统功能。有关更多信息，请参阅 GitHub 上的适用于 Amazon S3 的 Mountpoint 的文件系统行为。需要更丰富的文件系统语义的客户应该考虑其他 AWS 文件服务，例如 Amazon Elastic File System（Amazon EFS）或 Amazon FSx。

3. 使用 `umount` 命令卸载您的存储桶。此命令卸载您的 S3 存储桶并退出 Mountpoint。

   要使用以下示例命令，请将 *~/mnt* 替换为主机上挂载 S3 存储桶的目录。

   ```
   umount ~/mnt
   ```

   > **Note**
   >
   > 要获取此命令的选项列表，请运行 `umount --help`。

有关 Mountpoint 配置的其他详细信息，请参阅 GitHub 上的 S3 存储桶配置和文件系统配置。

在 Mountpoint 中配置缓存

在使用适用于 Amazon S3 的 Mountpoint 时，可以将它配置为在 Amazon EC2 实例存储或附加的 Amazon EBS 卷上缓存 S3 存储桶中最近访问的数据。缓存此数据有助于提高性能并降低重复数据访问的成本。在 Mountpoint 中缓存非常适合重复读取相同数据的使用场景，这些数据在多次读取期间不会发生更改。例如，可以将缓存用于需要多次读取训练数据集的机器学习训练作业，以提高模型精度。

挂载 S3 存储桶时，可以选择通过标志启用缓存。可以配置数据缓存的位置和大小以及元数据在缓存中的保留时间。在挂载存储桶并启用缓存后，Mountpoint 会在配置的缓存位置创建一个空的子目录（如果该子目录尚不存在）。首次挂载存储桶和卸载存储桶时，Mountpoint 会删除缓存位置的内容。有关在 Mountpoint 中配置和使用缓存的更多信息，请参阅 GitHub 上的适用于 Amazon S3 的 Mountpoint 缓存配置。

挂载 S3 存储桶时，可以使用 `--cache` *CACHE_PATH* 标志启用缓存。在以下示例中，将 *CACHE_PATH* 替换为要在其中缓存数据的目录的文件路径。将 *DOC-EXAMPLE-BUCKET* 替换为 S3 存储桶的名称，并将 *~/mnt* 替换为主机上您希望在其中挂载 S3 存储桶的目录。

```
mkdir ~/mnt
mount-s3 --cache CACHE_PATH DOC-EXAMPLE-BUCKET ~/mnt
```

> **Important**
>
> 如果启用缓存，Mountpoint 会将 S3 存储桶中未加密的对象内容保存在挂载时配置的缓存位置。建议您限制对数据缓存位置的访问以保护您的数据。

Mountpoint 故障排除

适用于 Amazon S3 的 Mountpoint 由 AWS Support 提供支持。如果您需要帮助，请联系 AWS Support 中心。

您也可以在 GitHub 上查看并提交 Mountpoint 问题。

如果您在该项目中发现潜在的安全问题，我们要求您通过我们的漏洞报告页面通知 AWS 安全部门。不要创建公开的 GitHub 问题。

如果您的应用程序在使用 Mountpoint 时出现异常行为，则可以检查日志信息以诊断问题。

日志记录

默认情况下，Mountpoint 会将高严重性日志信息发送到 syslog。

要查看最新的 Linux 发行版（包括 Amazon Linux）上的日志，请运行以下 `journald` 命令：

```
journalctl -e SYSLOG_IDENTIFIER=mount-s3
```

在其他 Linux 系统上，`syslog` 条目很可能写入到文件（例如 `/var/log/syslog`）中。

您可以使用这些日志对应用程序进行故障排除。例如，如果您的应用程序尝试覆盖现有文件，操作将失败，并且您将在日志中看到类似于以下内容的行：

```
[WARN] open{req=12 ino=2}: mountpoint_s3::fuse: open failed: inode error: inode 2 (full
 key "README.md") is not writable
```

有关更多信息，请参阅 GitHub 上的适用于 Amazon S3 的 Mountpoint 的日志记录。

使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输

Amazon S3 Transfer Acceleration 是一项存储桶级别功能，可在您的客户端和 S3 存储桶之间实现快速、轻松、安全的远距离文件传输。Transfer Acceleration 旨在优化从世界各地传入 S3 存储桶的传输速度。Transfer Acceleration 利用 Amazon CloudFront 中的全球分布式边缘站点。当数据到达某个边缘站点时，数据会被经过优化的网络路径路由至 Amazon S3。

使用 Transfer Acceleration 时，可能会收取额外的数据传输费用。有关定价的更多信息，请参阅 Amazon S3 定价。

为什么要使用 Transfer Acceleration？

您可能出于各种原因需要对存储桶使用 Transfer Acceleration：

- 您位于全球各地的客户需要上传到集中式存储桶。
- 您定期跨大洲传输数 GB 至数 TB 数据。
- 您在上传到 Amazon S3 时无法充分利用 Internet 上的所有可用带宽。

有关何时使用 Transfer Acceleration 的更多信息，请参阅 Amazon S3 常见问题解答。

使用 Transfer Acceleration 的要求

在 S3 存储桶上使用 Transfer Acceleration 时，需要执行以下操作：

- 仅虚拟托管样式请求支持 Transfer Acceleration。有关虚拟托管类型请求的更多信息，请参阅使用 REST API 提出请求。
- 用于 Transfer Acceleration 的存储桶的名称必须符合 DNS 标准，且不得包含句点（“.”）。
- 必须在存储桶上启用 Transfer Acceleration。有关更多信息，请参阅 启用和使用 S3 Transfer Acceleration。

  在对存储桶启用 Transfer Acceleration 后，可能需要最多 20 分钟的时间才能加快向存储桶传输数据的速度。

> **Note**
>
> 位于以下区域的存储桶目前支持 Transfer Acceleration：

  - 亚太地区（东京）(ap-northeast-1)
  - 亚太地区（首尔）(ap-northeast-2)
  - 亚太地区（孟买）(ap-south-1)
  - 亚太地区（新加坡）(ap-southeast-1)
  - 亚太地区（悉尼）(ap-southeast-2)
  - 加拿大（中部）(ca-central-1)
  - 欧洲地区（法兰克福）(eu-central-1)
  - 欧洲地区（爱尔兰）(eu-west-1)
  - 欧洲（伦敦）(eu-west-2)
  - 欧洲地区（巴黎）（eu-west-3）
  - 南美洲（圣保罗）（sa-east-1）
  - 美国东部（弗吉尼亚州北部）(us-east-1)
  - 美国东部（俄亥俄州）(us-east-2)
  - 美国西部（加利福尼亚北部）(us-west-1)
  - 美国西部（俄勒冈州）(us-west-2)

- 要访问已启用 Transfer Acceleration 的存储桶，您必须使用端点 *bucketname*.s3-accelerate.amazonaws.com。或者使用双堆栈端点 *bucketname*.s3-accelerate.dualstack.amazonaws.com，以通过 IPv6 连接至启用的存储桶。您可以继续使用常规端点进行标准数据传输。
- 您必须是存储桶拥有者才能设置传输加速状态。存储桶拥有者可以向其他用户分配权限，使他们能够对存储桶设置加速状态。s3:PutAccelerateConfiguration 权限允许用户对存储桶启用或禁用 Transfer Acceleration。s3:GetAccelerateConfiguration 权限允许用户返回存储桶的 Transfer Acceleration 状态，即 Enabled 或 Suspended.。

以下各节介绍了如何开始使用 Amazon S3 Transfer Acceleration 传输数据。

主题

- 开始使用 Amazon S3 Transfer Acceleration
- 启用和使用 S3 Transfer Acceleration
- 使用 Amazon S3 Transfer Acceleration 速度比较工具

开始使用 Amazon S3 Transfer Acceleration

您可以使用 Amazon S3 Transfer Acceleration 在您的客户端和 S3 存储桶之间进行快速、轻松、安全的远距离文件传输。Transfer Acceleration 使用 Amazon CloudFront 中的全球分布式边缘站点。当数据到达某个边缘站点时，数据会被经过优化的网络路径路由至 Amazon S3。

要开始使用 Amazon S3 Transfer Acceleration，请执行以下步骤：

1. 在存储桶上启用 Transfer Acceleration

   您可以通过以下任一方式对存储桶启用 Transfer Acceleration：

   - 使用 Amazon S3 控制台。
   - 使用 REST API PUT Bucket 加速操作。
   - 使用 AWS CLI 和 AWS SDK。有关更多信息，请参阅 使用 AWS SDK 通过 Amazon S3 进行开发。

   有关更多信息，请参阅 启用和使用 S3 Transfer Acceleration。

   > **Note**
   >
   > 对于要使用传输加速的存储桶，存储桶名称必须符合 DNS 命名要求，且不得包含句点（"."）。

2. 在启用加速的存储桶之间传输数据

   使用以下 s3-accelerate 端点域名之一：

   - 要访问启用加速的存储桶，请使用 *bucketname*.s3-accelerate.amazonaws.com。
   - 要通过 IPv6 访问启用了加速的存储桶，请使用*bucketname*.s3-accelerate.dualstack.amazonaws.com。

     Amazon S3 双堆栈端点支持通过 IPv6 和 IPv4 向 S3 存储桶发出请求。Transfer Acceleration 双堆栈端点仅可使用端点名称的虚拟托管类型。有关更多信息，请参阅通过 IPv6 发出请求入门和使用 Amazon S3 双堆栈端点。

   > **Note**
   >
   > 您的数据传输应用程序必须使用以下两种类型的端点之一来访问存储桶以加快数据传输：.s3-accelerate.amazonaws.com，或 .s3-

`accelerate.dualstack.amazonaws.com` 用于双栈端点。如果您想使用标准数据传输，则可以继续使用常规端点。

在启用 Transfer Acceleration 功能后，您可以将 Amazon S3 PUT 对象和 GET 对象请求指向 `s3-accelerate` 端点域名。例如，假设您当前有一个 REST API 应用程序使用 PUT 对象（该对象在 PUT 请求中使用主机名 `mybucket.s3.us-east-1.amazonaws.com`）。要加速 PUT，请将请求中的主机名更改为 `mybucket.s3-accelerate.amazonaws.com`。要重新使用标准上传速度，请将名称更改回 `mybucket.s3.us-east-1.amazonaws.com`。

启用 Transfer Acceleration 后，最多 20 分钟后即可实现性能提升。但是，一旦启用 Transfer Acceleration，加速端点将随即可用。

您可以在 AWS CLI、AWS SDK 和其他向 Amazon S3 传入数据和从 Amazon S3 传出数据的工具中使用加速端点。如果您使用 AWS SDK，则某些受支持的语言会使用加速端点客户端配置标记，这样一来，您便无需显式将 Transfer Acceleration 的端点设置为 *`bucketname`*`.s3-accelerate.amazonaws.com`。有关如何使用加速端点客户端配置标记的示例，请参阅启用和使用 S3 Transfer Acceleration。

您可以通过传输加速端点使用所有的 Amazon S3 操作，除以下情况以外：

- GET 服务（列出存储桶）
- PUT Bucket（创建存储桶）
- DELETE Bucket

此外，Amazon S3 Transfer Acceleration 不支持使用 PUT Object - Copy 进行跨区域复制。

启用和使用 S3 Transfer Acceleration

您可以使用 Amazon S3 Transfer Acceleration 在您的客户端和 S3 存储桶之间进行快速、安全的远距离文件传输。您可以使用 S3 控制台、AWS Command Line Interface（AWS CLI）、API 或 AWS SDK 启用 Transfer Acceleration。

本节提供了有关如何对存储桶启用 Amazon S3 Transfer Acceleration 和对启用的存储桶使用加速端点的示例。

有关 Transfer Acceleration 要求的更多信息，请参阅使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输。

使用 S3 控制台

> **Note**
>
> 如果要比较加快的上传速度与未加快的上传速度，请打开 Amazon S3 Transfer Acceleration 速度比较工具。
> 此速度比较工具使用分段上传来将文件从浏览器传输到各种使用和未使用 Amazon S3 Transfer Acceleration 的 AWS 区域。您可以比较直接上传和按区域传输加速上传的上传速度。

为 S3 存储桶启用传输加速

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，选择要为其启用加速传输的存储桶的名称。
3. 选择属性。
4. 在 Transfer acceleration (传输加速) 下，选择 Edit (编辑)。
5. 选择 Enable（启用），然后选择 Save changes（保存更改）。

要访问加速数据传输

1. 在 Amazon S3 为您的存储桶启用传输加速后，查看存储桶的 Properties（属性）选项卡。
2. 在 Transfer acceleration（传输加速）下，Accelerated endpoint（加速端点）显示存储桶的传输加速端点。使用此端点访问与存储桶之间的加速数据传输。

   如果您暂停传输加速，加速端点不再起作用。

使用 AWS CLI

以下是用于 Transfer Acceleration 的 AWS CLI 命令的示例。有关设置 AWS CLI 的说明，请参阅 使用 AWS CLI 进行 Amazon S3 开发。

在存储桶上启用 Transfer Acceleration

使用 AWS CLI put-bucket-accelerate-configuration 命令对存储桶启用或暂停 Transfer Acceleration。

以下示例设置 `Status=Enabled` 以对存储桶启用 Transfer Acceleration。您用 `Status=Suspended` 来暂停 Transfer Acceleration。

Example

```
$ aws s3api put-bucket-accelerate-configuration --bucket bucketname --accelerate-
configuration Status=Enabled
```

使用 Transfer Acceleration

您可以将 s3 和 s3api AWS CLI 命令发出的所有 Amazon S3 请求定向到加速端点：`s3-accelerate.amazonaws.com`。为此，请在 AWS Config 文件的配置文件中将配置值 `use_accelerate_endpoint` 设置为 `true`。必须对存储桶启用 Transfer Acceleration 才能使用加速端点。

所有请求都使用存储桶寻址的虚拟风格发送：`my-bucket.s3-accelerate.amazonaws.com`。不会将任何 `ListBuckets`、`CreateBucket` 和 `DeleteBucket` 请求发送到加速端点，因为该端点不支持这些操作。

有关 `use_accelerate_endpoint` 的更多信息，请参阅《AWS CLI 命令参考》中的 AWS CLI S3 配置。

以下示例在默认配置文件中将 `use_accelerate_endpoint` 设置为 `true`。

Example

```
$ aws configure set default.s3.use_accelerate_endpoint true
```

如果您需要对某些 AWS CLI 命令使用加速端点，但不对其他此类命令使用加速端点，则可使用以下两种方法中的任一方法：

- 通过将 `--endpoint-url` 参数设置为 `https://s3-accelerate.amazonaws.com`，来对任何 s3 或 s3api 命令使用加速端点。
- 在 AWS Config 文件中设置单独的配置文件。例如，创建一个将 `use_accelerate_endpoint` 设置为 `true` 的配置文件和一个不设置 `use_accelerate_endpoint` 的配置文件。在运行一条命令时，根据是否需要使用加速端点来指定要使用的配置文件。

将对象上传到已启用 Transfer Acceleration 的存储桶

以下示例通过使用已配置为使用加速端点的默认配置文件来将文件上传到已启用 Transfer Acceleration 的存储桶。

Example

```
$ aws s3 cp file.txt s3://bucketname/keyname --region region
```

以下示例通过使用 `--endpoint-url` 参数指定加速端点来将文件上传到已启用 Transfer Acceleration 的存储桶。

Example

```
$ aws configure set s3.addressing_style virtual
$ aws s3 cp file.txt s3://bucketname/keyname --region region --endpoint-url https://s3-accelerate.amazonaws.com
```

使用 AWS SDK

以下是使用 Transfer Acceleration 通过 AWS SDK 将对象上传到 Amazon S3 的示例。一些 AWS SDK 支持的语言（例如，Java 和 .NET）使用加速端点客户端配置标记，这样一来，您便无需显式将 Transfer Acceleration 的端点设置为 *bucketname*.s3-accelerate.amazonaws.com。

Java

Example

以下示例演示如何使用加速端点将对象上传到 Amazon S3。本示例执行以下操作：

- 创建配置为使用加速端点的 `AmazonS3Client`。客户端访问的所有存储桶都必须已启用 Transfer Acceleration。
- 对指定存储桶启用 Transfer Acceleration。仅当您指定的存储桶尚未启用 Transfer Acceleration 时，此步骤才是必需的。
- 验证是否为指定存储桶启用了传输加速。
- 使用存储桶的加速端点将新对象上传到指定存储桶。

有关使用 Transfer Acceleration 的更多信息，请参阅开始使用 Amazon S3 Transfer Acceleration。有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketAccelerateConfiguration;
import com.amazonaws.services.s3.model.BucketAccelerateStatus;
import com.amazonaws.services.s3.model.GetBucketAccelerateConfigurationRequest;
import com.amazonaws.services.s3.model.SetBucketAccelerateConfigurationRequest;

public class TransferAcceleration {
    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";

        try {
            // Create an Amazon S3 client that is configured to use the accelerate
 endpoint.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .enableAccelerateMode()
                    .build();

            // Enable Transfer Acceleration for the specified bucket.
            s3Client.setBucketAccelerateConfiguration(
                    new SetBucketAccelerateConfigurationRequest(bucketName,
                            new BucketAccelerateConfiguration(
                                    BucketAccelerateStatus.Enabled)));

            // Verify that transfer acceleration is enabled for the bucket.
            String accelerateStatus = s3Client.getBucketAccelerateConfiguration(
                    new GetBucketAccelerateConfigurationRequest(bucketName))
                    .getStatus();
            System.out.println("Bucket accelerate status: " + accelerateStatus);

            // Upload a new object using the accelerate endpoint.
            s3Client.putObject(bucketName, keyName, "Test object for transfer
 acceleration");
            System.out.println("Object \"" + keyName + "\" uploaded with transfer
 acceleration.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
```

```
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

以下示例说明如何使用 AWS SDK for .NET 对存储桶启用 Transfer Acceleration。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TransferAccelerationTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            EnableAccelerationAsync().Wait();
        }

        static async Task EnableAccelerationAsync()
        {
                try
                {
                    var putRequest = new PutBucketAccelerateConfigurationRequest
                    {
```

```
                        BucketName = bucketName,
                        AccelerateConfiguration = new AccelerateConfiguration
                        {
                            Status = BucketAccelerateStatus.Enabled
                        }
                    };
                    await
 s3Client.PutBucketAccelerateConfigurationAsync(putRequest);

                    var getRequest = new GetBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName
                    };
                    var response = await
 s3Client.GetBucketAccelerateConfigurationAsync(getRequest);

                    Console.WriteLine("Acceleration state = '{0}' ",
 response.Status);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    Console.WriteLine(
                        "Error occurred. Message:'{0}' when setting transfer
 acceleration",
                        amazonS3Exception.Message);
                }
        }
    }
}
```

在将对象上传到启用了 Transfer Acceleration 的存储桶时，可指定在创建客户端时使用加速端点。

```
var client = new AmazonS3Client(new AmazonS3Config
            {
                RegionEndpoint = TestRegionEndpoint,
                UseAccelerateEndpoint = true
            }
```

Javascript

有关通过使用适用于 JavaScript 的 AWS SDK 来启用 Transfer Acceleration 的示例，请参阅《适用于 JavaScript 的 AWS SDK API 参考》中的调用 putBucketAccelerateConfiguration 操作。

Python (Boto)

有关通过使用适用于 Python 的 SDK 启用 Transfer Acceleration 的示例，请参阅《AWS SDK for Python（Boto3）API 参考》中的 put_bucket_accelerate_configuration。

Other

有关使用其他 AWS SDK 的信息，请参阅示例代码和库。

使用 REST API

使用 REST API `PutBucketAccelerateConfiguration` 操作在现有存储桶上启用加速配置。

有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketAccelerateConfiguration。

使用 Amazon S3 Transfer Acceleration 速度比较工具

您可以使用 Amazon S3 Transfer Acceleration 速度比较工具来比较各个 Amazon S3 区域内加快的上传速度和未加快的上传速度。此速度比较工具使用分段上传来将文件从浏览器传输到各种使用和未使用 Transfer Acceleration 的 Amazon S3 区域。

可使用以下任一方法访问此速度比较工具：

- 将以下 URL 复制到浏览器窗口中，并分别将 *region* 和 *yourBucketName* 替换为您使用的 AWS 区域（例如 us-west-2）和要评估的存储桶的名称：

  ```
  https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/
  accelerate-speed-comparsion.html?
  region=region&origBucketName=yourBucketName
  ```

  有关 Amazon S3 支持的区域列表，请参阅《AWS 一般参考》中的 Amazon S3 端点和限额。
- 使用 Amazon S3 控制台。

使用申请方付款存储桶进行存储传输和使用

通常，存储桶拥有者将支付与他们的存储桶相关联的所有 Amazon S3 存储和数据传输费用。但是，您可以将存储桶配置为申请方付款存储桶。使用申请方付款存储桶时，申请方（而不是存储桶拥有者）将支付请求和从存储桶下载数据的费用。存储桶拥有者将始终支付存储数据的费用。

通常情况下，当您想共享数据，而又不希望产生与访问数据等其他操作相关联的费用时，您可以将存储桶配置为申请方付款存储桶。例如，当提供大型数据集（如邮政编码目录、参考数据、地理空间信息或网络爬取数据）时，您可能会使用申请方付款存储桶。

> **Important**
>
> 如果您在存储桶上启用了申请方付款，则不允许匿名访问该存储桶。

您必须对涉及申请方付款存储桶的所有请求进行身份验证。请求身份验证使 Amazon S3 能够识别申请方对申请方付款存储桶的使用并对其收费。

当请求者在做出其请求前担任 AWS Identity and Access Management (IAM) 角色时，该角色所属的账户将负责处理此请求。有关 IAM 角色的更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

在将桶配置为申请方付款桶后，申请方必须表明他们了解请求和数据下载将产生费用。为了表明他们接受这些费用，申请方必须在 DELETE、GET、HEAD、POST 和 PUT 请求的 API 请求中包含 `x-amz-request-payer` 作为标头，或者在其 REST 请求中添加 `RequestPayer` 参数。对于 CLI 请求，申请方可以使用 `--request-payer` 参数。

Example – 删除对象时使用申请方付款

要使用以下 DeleteObjectVersion API 示例，请将 *`user input placeholders`* 替换为您自己的信息。

```
DELETE /Key+?versionId=VersionId HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-mfa: MFA
x-amz-request-payer: RequestPayer
x-amz-bypass-governance-retention: BypassGovernanceRetention
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

如果申请方使用 RestoreObject API 还原对象，则只要请求中包含 `x-amz-request-payer` 标头或 `RequestPayer` 参数，就会支持申请方付款；但是，申请方只需支付请求的费用。检索费用由桶拥有者支付。

申请方付款存储桶不支持以下内容：

- 匿名申请
- SOAP 请求

- 将申请方付款存储桶用作最终用户日志记录的目标存储桶，反之亦然。但是，当目标存储桶不是申请方付款存储桶时，您可以在申请方付款存储桶上开启最终用户日志记录。

申请方付款的费用支付方式

成功的申请方付款请求费用简单明了：申请方支付数据传输和请求方面的费用；存储桶拥有者支付数据存储方面的费用。但是，在以下条件下会对存储桶拥有者收取请求费用：

- 该请求返回 `AccessDenied`（HTTP `403 Forbidden`）错误，并且该请求是在存储桶拥有者的个人 AWS 账户或 AWS 组织内发起的。
- 请求是 SOAP 请求。

有关申请方付款的更多信息，请参阅以下主题。

主题

- 在存储桶上配置申请方付款
- 使用 REST API 检索 requestPayment 配置
- 从申请方付款桶中下载对象

在存储桶上配置申请方付款

您可以将 Amazon S3 存储桶配置为申请方付款存储桶，以便请求者而不是存储桶拥有者支付请求和数据下载费用。

本节提供了演示如何使用控制台和 REST API 在 Amazon S3 存储桶上配置申请方付款的示例。

使用 S3 控制台

为 S3 存储桶启用申请方付款

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，选择要为其启用申请方付款的存储桶的名称。
3. 选择属性。
4. 在 Requester pays (申请方付款) 下，选择 Edit (编辑)。
5. 选择 Enable (启用)，然后选择 Save changes (保存更改)。

Amazon S3 为存储桶启用申请方付款，并显示 Bucket overview（存储桶概述）。在 Requester pays (申请方付款) 下，您将看到 Enabled (已启用)。

使用 REST API

只有存储桶拥有者才能将存储桶的 `RequestPaymentConfiguration.payer` 配置值设置为 `BucketOwner`（默认值）或 `Requester`。设置 `requestPayment` 资源是可选的。默认情况下，存储桶不是申请方付款存储桶。

要将申请方付款存储桶恢复为常规存储桶，请使用值 `BucketOwner`。通常情况下，在将数据上传到 Amazon S3 存储桶时，您将使用 `BucketOwner`，然后将值设置为 `Requester`，才能在该存储桶中发布对象。

设置 requestPayment 的步骤

- 使用 PUT 请求在指定存储桶上将 `Payer` 值设置为 `Requester`。

```
PUT ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Content-Length: 173
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]

<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

如果请求成功，Amazon S3 将返回类似于以下内容的请求。

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
x-amz-request-charged:requester
```

您只能在存储桶级别设置申请方付款。您无法为存储桶中的特定对象设置申请方付款。

您可以随时将存储桶配置为 BucketOwner 或 Requester。但是，新配置值可能需要几分钟才能生效。

> **Note**
>
> 在将存储桶配置为申请方付款之前，分发预签名 URL 的存储桶拥有者应当再三考虑，尤其是在 URL 的生命周期非常长时更应如此。在每次申请方使用预签名 URL（使用存储桶拥有者的凭证）时，会向存储桶拥有者收取费用。

使用 REST API 检索 requestPayment 配置

您可以通过请求资源 Payer 来确定在存储桶上设置的 requestPayment 值。

返回 requestPayment 资源的步骤

- 使用 GET 请求来获取 requestPayment 资源，如以下请求所示。

```
GET ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

如果请求成功，Amazon S3 将返回类似于以下内容的请求。

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Type: [type]
Content-Length: [length]
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

此响应显示 payer 值已设置为 Requester。

从申请方付款桶中下载对象

因为会向申请方收取从申请方付款存储桶下载数据的费用，因此请求必须包含特殊参数 x-amz-request-payer，该参数确认申请方了解将向他们收取下载费用。要在申请方付款存储桶中访问对象，请求必须包含以下内容之一。

- 对于 DELETE、GET、HEAD、POST 和 PUT 请求，在标头中包含 x-amz-request-payer : requester
- 对于已签名的 URL，需在请求中包括 x-amz-request-payer=requester

如果请求成功且已向申请方收取费用，则响应将包括标头 x-amz-request-charged:requester。如果请求中没有 x-amz-request-payer，Amazon S3 将返回 403 错误并向存储桶拥有者收取请求的费用。

> **Note**
>
> 存储桶拥有者无需将 x-amz-request-payer 添加到他们的请求。
> 确保在您的签名计算中包含 x-amz-request-payer 及其值。有关更多信息，请参阅构建 CanonicalizedAmzHeaders 元素。

使用 REST API

从申请方付款存储桶下载对象的步骤

- 使用 GET 请求从申请方付款存储桶下载对象，如以下请求所示。

```
GET / [destinationObject] HTTP/1.1
Host: [BucketName].s3.amazonaws.com
x-amz-request-payer : requester
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

如果 GET 请求成功且已向申请方收取费用，则响应将包括 x-amz-request-charged:requester。

Amazon S3 可以为尝试从申请方付款存储桶获取对象的请求返回 Access Denied 错误。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的错误响应。

使用 AWS CLI

要使用 AWS CLI 从申请方付款存储桶下载对象，请指定 `--request-payer requester` 包含在您的 `get-object` 请求中。有关更多信息，请参阅《AWS CLI 参考》中的 get-object。

存储桶配额、限制和局限性

Amazon S3 存储桶归创建它的 AWS 账户 所有。存储桶所有权不可转移到其他账户。

存储桶配额限制

默认情况下，您可以在每个 AWS 账户 中创建多达 100 个存储桶。如果您需要更多存储桶，则可以通过提交限额提升请求，将账户的存储桶限额提高至最多 1000 个存储桶。无论您使用许多存储桶还是少量存储桶，性能都没有差异。

> **Note**
>
> 您无需为每个 AWS 区域提交多个限额提升请求。您的存储桶限额适用于您的 AWS 账户。

有关如何增加存储桶配额的信息，请参阅《Amazon Web Services 一般参考》中的 Amazon S3 endpoints and quotas。

对象和存储桶限制

没有最大存储桶大小，对存储桶中可存储的项目数也没有限制。您可以在单个存储桶中存储所有对象，也可以在多个存储桶中组织它们。但是，您无法从其他存储桶中创建存储桶。

存储桶命名限制

创建存储桶时，您可以选择存储桶名称和要在其中创建存储桶的 AWS 区域。创建存储桶后，便无法再更改其名称或区域。

为存储桶命名时，请选择与您或您的组织相关的名称。避免使用与其他人关联的名称。例如，您应避免在存储桶名称中使用 AWS 或 Amazon。

重复使用存储桶名称

如果存储桶为空，您可以将其删除。删除存储桶后，该名称可供重复使用。但是，删除存储桶后，您可能由于各种原因而无法重复使用该名称。

例如，当您删除存储桶并且名称变为可供重用时，另一个 AWS 账户可能会创建具有该名称的存储桶。此外，可能需要一段时间才能重复使用已删除的存储桶的名称。如果您想使用相同的存储桶名称，我们建议您不要删除该存储桶。

有关存储桶名称的更多信息，请参阅存储桶命名规则。

存储桶命名和自动创建的存储桶

如果您的应用程序自动创建了存储桶，请选择不会导致命名冲突的存储桶命名方案。确保存储桶名称已被使用时，您的应用程序逻辑会选择其他存储桶名称。

有关存储桶命名的更多信息，请参阅 存储桶命名规则。

存储桶操作

Amazon S3 的高可用性设计主要关注获取、放置、列出和删除操作。由于存储桶操作在集中的全球资源空间中进行，因此不建议在应用程序的高可用性代码路径上创建、删除或配置存储桶。最好是在单独的初始化或设置不常运行的例程时创建、删除或配置存储桶。

上传、下载和处理 Amazon S3 中的对象

要将数据存储在 Amazon S3 中，您需要使用称为存储桶和对象的资源。存储桶是对象的容器。对象指的是一个文件和描述该文件的任何元数据。

要将对象存储在 Amazon S3 中，您需要创建存储桶，然后将该对象上传到存储桶。当对象位于存储桶时，您可以将其打开、下载并将其复制。当您不再需要对象或存储桶时，可以清理这些资源。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

> **Important**
>
> 在 Amazon S3 控制台中，当您为对象选择 Open（打开）或 Download As（下载为）时，这些操作会创建预签名 URL。在五分钟的时间内，任何有权访问这些预签名 URL 的人都可以访问您的对象。有关使用预签名 URL 的更多信息，请参阅使用预签名 URL。

使用 Amazon S3，您只需按实际用量付费。有关 Amazon S3 特征和定价的更多信息，请参阅 Amazon S3。如果您是 Amazon S3 的新客户，则可以免费开始使用 Amazon S3。有关更多信息，请参阅 AWS 免费套餐。

主题

- 删除 Amazon S3 对象
- 组织、列出和处理对象
- 使用预签名 URL
- 使用 S3 对象 Lambda 转换对象

Amazon S3 对象概述

Amazon S3 是一个对象存储，它使用唯一性键值来存储任意数量的对象。您将这些对象存储在一个或多个存储桶中，每个对象的大小最多可达 5 TB。对象由以下内容组成：

密钥

分配给对象的名称。您可以使用对象键检索该对象。有关更多信息，请参阅使用对象元数据。

版本 ID

在存储桶中，键和版本 ID 唯一地标识对象。版本 ID 是 Amazon S3 在对象添加到存储桶时生成的字符串。有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

值

您正在存储的内容。

对象值可以是任意序列的字节。对象的大小范围是 0 到 5 TB。有关更多信息，请参阅 上传对象。

Metadata

一组名称值对，可用于存储有关对象的信息。您可以将元数据（称为用户定义的元数据）分配给 Amazon S3 中的对象。Amazon S3 也可以将用于管理对象的系统元数据指定给这些对象。有关更多信息，请参阅使用对象元数据。

子资源

Amazon S3 使用子资源机制存储特定于对象的其他信息。因为子资源从属于对象，因此它们始终与某些其他实体（如对象或存储桶）相关联。有关更多信息，请参阅 对象子资源。

访问控制信息

您可以控制对您在 Amazon S3 中存储的对象的访问。Amazon S3 支持基于资源的访问控制（例如访问控制列表（ACL）和存储桶策略）和基于用户的访问控制。有关访问控制的更多信息，请参阅以下内容：

- 访问控制

- Amazon S3 的身份和访问管理
- 配置 ACL

您的 Amazon S3 资源（例如存储桶和对象）在默认情况下是私有的。您必须显式授予权限，才能允许其他人访问这些资源。有关共享对象的更多信息，请参阅使用预签名 URL 共享对象。

标签

您可以使用标签对存储的对象进行分类，进行访问控制或成本分配。有关更多信息，请参阅 使用标签对存储进行分类。

对象子资源

Amazon S3 将定义一组与存储桶和对象相关联的子资源。子资源是对象的从属项。这意味着子资源不独立存在。它们始终与某些其他实体（如对象或存储桶）相关联。

下表列出了与 Amazon S3 对象相关联的子资源。

子资源	描述
acl	包含可以识别被授权者和所授予的权限的授权列表。创建对象时，`acl` 将识别可以完全控制对象的对象所有者。您可以检索对象 ACL 或将其替换为更新的授权列表。对 ACL 的任何更新都需要您替换现有 ACL。有关 ACL 的更多信息，请参阅 访问控制列表 (ACL) 概述。

创建对象键名称

对象键（或键名称）唯一标识 Amazon S3 存储桶中的对象。对象元数据是一组名称值对。有关对象元数据的更多信息，请参阅 使用对象元数据。

创建对象时，要指定键名，它在存储桶中唯一地标识该对象。例如，在 Amazon S3 控制台上，在突出显示存储桶时，将显示存储桶中的对象的列表。这些名称是对象键。对象键名称是 Unicode 字符序列，它采用 UTF-8 编码，长度最大为 1,024 字节。对象键名称区分大小写。

> **Note**
>
> 对于虚拟托管类型请求，不支持值为“soap”的对象键名称。对于使用“soap”的对象键名称值，必须改用路径类型 URL。

Amazon S3 数据模型是一种扁平结构：您创建一个存储桶，存储桶存储对象。不存在子存储桶或子文件夹的层次结构。但您可以使用键名前缀和分隔符推断逻辑层次结构（如同 Amazon S3 控制台一样）。Amazon S3 控制台支持文件夹的概念。有关如何从 Amazon S3 控制台编辑元数据的更多信息，请参阅在 Amazon S3 控制台中编辑对象元数据。

假设您的存储桶（admin-created）包含具有以下对象键的四个对象：

```
Development/Projects.xls
```

```
Finance/statement1.pdf
```

```
Private/taxdocument.pdf
```

```
s3-dg.pdf
```

控制台使用键名前缀 (Development/、Finance/ 和 Private/) 和分隔符 (“/”) 呈现文件夹结构。s3-dg.pdf 键没有前缀，因此其对象直接在存储桶的根级别出现。如果您打开 Development/ 文件夹，可以看到 Projects.xlsx 对象。

- Amazon S3 支持存储桶和对象且没有层次结构。但是，通过在对象键名中使用前缀和分隔符，Amazon S3 控制台和 AWS SDK 可以推断层次结构并引入文件夹的概念。
- Amazon S3 控制台通过创建以文件夹前缀和分隔符值作为键的零字节对象来实现文件夹对象创建。这些文件夹对象不会显示在控制台中。否则，它们的行为与任何其他对象一样，可以通过 REST API、AWS CLI 和 AWS SDK 进行查看和操作。

对象键命名准则

您可以在对象键名中使用任意 UTF-8 字符。但是，在键名中使用某些字符可能导致一些应用程序和协议出现问题。以下指导原则有助于最大程度符合 DNS、Web 安全字符、XML 分析器和其他 API 的要求。

安全字符

以下字符集通常可安全地用于键名。

Alphanumeric characters	• 0-9 • a-z • A-Z
Special characters	• 感叹号 (!)

- 连字符 (-)
- 下划线 (_)
- 句点 (.)
- 星号 (*)
- 单引号 (')
- 左括号 (()
- 右括号 ())

以下是有效对象键名的示例：

- `4my-organization`
- `my.great_photos-2014/jan/myvacation.jpg`
- `videos/2014/birthday/video1.wmv`

> **Note**
>
> 使用 Amazon S3 控制台下载的键名以句点“.”结尾的对象将从下载对象的键名中删除句点“.”。要下载键名以句点“.”结尾并保留在下载对象中的对象，必须使用 AWS Command Line Interface（AWS CLI）、AWS SDK 或 REST API。
> 此外，请注意以下前缀限制：
>
> - 对于前缀为“./”的对象，必须使用 AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 上传或下载。您不能使用 Amazon S3 控制台。
> - 对于前缀为“../”的对象，不能使用 AWS Command Line Interface (AWS CLI) 或 Amazon S3 控制台上传。

可能需要特殊处理的字符

键名中的以下字符可能需要另外进行代码处理，并且可能需要以十六进制形式在 URL 中编码或引用。其中部分字符是不可打印的字符，浏览器可能无法处理它们，这也需要特殊处理：

- 表示和的符号 (“&”)
- 美元 (“$”)
- ASCII 字符范围 00–1F 十六进制 (0–31 十进制) 和 7F (127 十进制)

- “At”符号 (“@”)
- 等于 (“=”)
- 分号 (“;”)
- 正斜杠 ("/")
- 冒号 (“:”)
- 加号 (“+”)
- 空格 – 大量连续空格可能会在某些使用情形中丢失（特别是多个空格）
- 逗号 (“,”)
- 问号 (“?”)

要避免的字符

我们建议不要在键名中使用以下字符，因为这些字符需要进行大量特殊的字符处理，才能在所有应用程序间保持一致性。

- 反斜杠（“\”）
- 左大括号 (“{”)
- 不可打印的 ASCII 字符 (128–255 十进制字符)
- 插入符号 (“^”)
- 右大括号 (“}”)
- 百分比字符 (“%”)
- 重音符/反勾号 (“`”)
- 右方括号 (“]”)
- 引号
- “大于”符号 (“>”)
- 左方括号 (“[”)
- 波浪字符 (“~”)
- “小于”符号（“<”）
- “井号”字符 (“#”)
- 竖线 (“|”)

XML 相关的对象键约束

正如关于行尾处理的 XML 标准所指定的那样，所有 XML 文本都被规范化，这样单个回车符（ASCII 代码 13）和回车符紧跟换行符（ASCII 代码 10）将替换为单个换行符。为了确保正确解析 XML 请求中的对象键，当将回车符和其他特殊字符插入 XML 标签时，必须使用等效的 XML 实体代码替换回车符和其他特殊字符。以下是此类特殊字符及其等效实体代码的列表：

- ' 用作 '
- " 用作 "
- & 用作 &
- < 用作 <
- > 用作 >
- \r 用作
 或
- \n 用作
 或

Example

以下示例说明了使用 XML 实体代码替换回车的情况。此 DeleteObjects 请求将删除带有 key 参数的对象：/some/prefix/objectwith\rcarriagereturn（其中 \r 是回车）。

```
<Delete xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Object>
    <Key>/some/prefix/objectwith&#13;carriagereturn</Key>
  </Object>
</Delete>
```

使用对象元数据

您可以在上传对象时在 Amazon S3 中设置对象元数据。对象元数据是一组名称值对。上传对象后，您将无法修改对象元数据。修改对象元数据的唯一方式是创建对象的副本并设置元数据。

创建对象时，还要指定键名，它在存储桶中唯一地标识该对象。对象键（或键名称）唯一标识 Amazon S3 存储桶中的对象。有关更多信息，请参阅 创建对象键名称。

Amazon S3 中有两种元数据：系统定义的元数据和用户定义的元数据。以下各节提供了有关系统定义和用户定义的元数据的更多信息。有关使用 Amazon S3 控制台编辑元数据的更多信息，请参阅在 Amazon S3 控制台中编辑对象元数据。

系统定义的对象元数据

对存储桶中存储的每个对象，Amazon S3 将保留一组系统元数据。Amazon S3 将根据需要处理此系统元数据。例如，Amazon S3 将保留对象创建日期和大小元数据，并将此信息用作对象管理的一部分。

有两种类别的系统元数据：

- 受系统控制 – 元数据（例如对象创建日期）受系统控制，这意味着仅 Amazon S3 可以修改该值。
- 受用户控制 – 其他系统元数据（例如，为对象配置的存储类以及对象是否已启用服务器端加密）是您可以控制其值的系统元数据的示例。如果您的存储桶配置为网站，有时您可能想要将页面请求重定向到其他页面或外部 URL。在这种情况下，网页就是您的存储桶中的对象。Amazon S3 将存储页面重定向值作为您可以控制其值的系统元数据。

  创建对象时，您可以配置这些系统元数据项目的值或在需要时更新这些值。有关存储类的更多信息，请参阅 使用 Amazon S3 存储类。

  Amazon S3 使用 AWS KMS 密钥加密您的 Amazon S3 对象。AWS KMS 只加密对象数据。校验和与指定的算法一起存储为对象的元数据的一部分。如果请求为对象进行服务器端加密，则校验和将以加密形式存储。有关服务器端加密的更多信息，请参阅 利用加密来保护数据。

> **Note**
>
> PUT 请求标头的大小限制为 8KB。在 PUT 请求标头中，系统定义的元数据的大小限制为 2KB。通过计算每个键和值的 US-ASCII 编码中的字节总数来测量系统定义的元数据的大小。

下表提供了系统定义的元数据列表以及您是否可以更新它。

名称	描述	用户是否可以修改该值？
`Date`	当前日期和时间。	否
`Cache-Control`	用于指定缓存策略的常规标头字段。	是
`Content-Disposition`	对象演示信息。	是

名称	描述	用户是否可以修改该值？
Content-Length	对象大小（以字节为单位）。	否
Content-Type	对象类型。	是
Last-Modified	对象创建日期或上次修改日期（以较晚者为准）。对于分段上传，对象的创建日期是开始分段上传的日期。	否
ETag	表示对象的特定版本的实体标签（ETag）。对于未作为分段上传上传、未加密或使用 Amazon S3 托管密钥 (SSE-S3) 的服务器端加密进行加密的对象，ETag 是数据的 MD5 摘要。	否
x-amz-server-side-encryption	一个标头，它指示是否为对象启用了服务器端加密，以及加密是使用 AWS Key Management Service（AWS KMS）密钥（SSE-KMS），还是使用 Amazon S3 托管式加密密钥（SSE-S3）。有关更多信息，请参阅 使用服务器端加密保护数据。	是
x-amz-checksum-crc32 ，x-amz-checksum-crc32c ，x-amz-checksum-sha1 ，x-amz-checksum-sha256	包含对象的校验和或摘要的标头。最多一次设置其中一个标头，具体取决于您指示 Amazon S3 使用的校验和算法。有关选择校验和算法的更多信息，请参阅检查对象完整性。	否
x-amz-version-id	对象版本。对存储桶启用版本控制时，Amazon S3 会为添加到存储桶的对象指定版本 ID。有关更多信息，请参阅 在 S3 存储桶中使用版本控制。	否
x-amz-delete-marker	一个布尔值标记，指示对象是否为删除标记。此标记仅在启用了版本控制的存储桶中使用。	否

名称	描述	用户是否可以修改该值？
x-amz-storage-class	用于存储对象的存储类。有关更多信息，请参阅 使用 Amazon S3 存储类。	是
x-amz-website-redirect-location	一个标头，用于将对于关联对象的请求重定向到同一个存储桶中的其他对象或外部 URL。有关更多信息，请参阅（可选）配置网页重定向。	是
x-amz-server-side-encryption-aws-kms-key-id	一个标头，表示用于加密对象的 AWS KMS 对称加密 KMS 密钥的 ID。此标头仅在 x-amz-server-side-encryption 标头存在且值为 aws:kms 时使用。	是
x-amz-server-side-encryption-customer-algorithm	一个标头，指示是否启用了具有客户提供的加密密钥的服务器端加密（SSE-C）。有关更多信息，请参阅 使用具有客户提供的密钥的服务器端加密（SSE-C）。	是
x-amz-tagging	对象的标签集。标签集必须编码为 URL 查询参数。	是

用户定义的对象元数据

上传对象时，您也可以将元数据指定给该对象。发送 PUT 或 POST 请求以创建对象时，您将以名称-值（键-值）对的形式提供此可选信息。如果使用 REST API 上传对象，可选的用户定义的元数据名称必须以 x-amz-meta- 开头，以与其他 HTTP 标头区分开来。使用 REST API 检索对象时，将返回此前缀。如果使用 SOAP API 上传对象则无需前缀。使用 SOAP API 检索对象时，无论您使用哪种 API 上传对象，都将删除前缀。

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 特征。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

通过 REST API 检索元数据时，Amazon S3 会将具有相同名称（不分大小写）的标头合并为逗号分隔的列表。如果某些元数据包含不可打印的字符，则不会返回它。但是，`x-amz-missing-meta` 标头将与不可打印的元数据条目的数量值一起返回。该 `HeadObject` 操作从对象检索元数据而不返回对象本身。如果您只对某个对象的元数据感兴趣，则此操作很有用。要使用 `HEAD`，您必须具有对象的 `READ` 访问权限。有关更多信息，请参阅 Amazon Simple Storage Service API 参考 中的 HeadObject。

用户定义的元数据是一组键值对。Amazon S3 以小写形式存储用户定义的元数据键。

Amazon S3 允许元数据值中有任意 Unicode 字符。

为避免与这些元数据值的显示有关的问题，如果使用 SOAP 或通过 `POST` 进行基于浏览器的上传，则当使用 REST 和 UTF-8 时，应遵守使用 US-ASCII 字符的规定。

在元数据值中使用非 US-ASCII 字符时，将检查提供的 Unicode 字符串中是否存在非 US-ASCII 字符。此类标头的值在存储前按照 RFC 2047 进行字符解码，并在返回前按照 RFC 2047 进行编码以确保邮件安全。如果字符串仅包含 US-ASCII 字符，则按原样显示。

示例如下：

```
PUT /Key HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.amazonaws.com
x-amz-meta-nonascii: ÄMÄZÕÑ S3

HEAD /Key HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.amazonaws.com
x-amz-meta-nonascii: =?UTF-8?B?w4PChE3Dg8KEWsODwpXDg8KRIFMz?=

PUT /Key HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3

HEAD /Key HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3
```

> **Note**
>
> PUT 请求标头的大小限制为 8KB。在 PUT 请求标头中，用户定义的元数据的大小限制为 2KB。通过计算每个键和值的 UTF-8 编码中的字节总数来测量用户定义的元数据的大小。

有关在上传对象后通过创建对象的副本、修改对象、替换旧对象或创建新版本来更改对象的元数据的信息，请参阅在 Amazon S3 控制台中编辑对象元数据。

在 Amazon S3 控制台中编辑对象元数据

您可以使用 Amazon S3 控制台编辑现有 S3 对象的元数据。在您上传对象时，Amazon S3 会设置一些元数据。例如，`Content-Length` 和 `Last-Modified` 是系统定义的对象元数据字段，用户无法修改。

您也可以在上传对象时设置一些元数据，并稍后在需求更改时对其进行编辑。例如，您可能有一组初始存储在 `STANDARD` 存储类中的对象。随着时间的推移，您可能不再需要这些数据具有高可用性。因此，您可以通过将 `GLACIER` 密钥的值从 `x-amz-storage-class` 更改为 `STANDARD` 来将存储类更改为 `GLACIER`。

> **Note**
>
> 在 Amazon S3 中编辑对象元数据时，请考虑以下问题：
>
> - 此操作将使用更新的设置和上次修改日期创建对象的副本。如果启用 S3 版本控制，则会创建对象的新版本，而现有对象将变为旧版本。如果未启用 S3 版本控制，则对象的新副本将替换原始对象。与更改属性的 IAM 角色关联的 AWS 账户还会成为新对象或（对象版本）的拥有者。
> - 要使用 Amazon S3 控制台为具有用户定义标签的对象编辑元数据，您还必须拥有 `s3:GetObjectTagging` 权限。如果您使用 Amazon S3 控制台为没有用户定义标签但大小超过 16 MB 的对象编辑元数据，您还必须拥有 `s3:GetObjectTagging` 权限。
>
>   如果目标存储桶策略拒绝 `s3:GetObjectTagging` 操作，则将更新对象的元数据，但将从对象中移除用户定义的标签，并且您将收到错误。
> - 编辑元数据会更新现有键名的值。
> - 无法使用控制台复制使用客户提供的加密密钥 (SSE-C) 加密的对象。您必须使用 AWS CLI、AWS SDK 或 Amazon S3 REST API。

> **Warning**
>
> 编辑文件夹的元数据时，请等待 `Edit metadata` 操作完成，然后再将新对象添加到文件夹。否则，也可能会编辑新对象。

以下主题介绍了如何使用 Amazon S3 控制台编辑对象的元数据。

编辑系统定义的元数据

您可以为 S3 对象配置某些（但并非全部）系统元数据。有关系统定义元数据的列表以及您能否修改其值的信息，请参阅系统定义的对象元数据。

编辑对象的系统定义的元数据

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 导航到您的 Amazon S3 存储桶或文件夹，然后选中要编辑其元数据的对象名称左侧的复选框。
3. 在 Actions（操作）菜单上，请选择 Edit actions（编辑操作），然后选择 Edit metadata（编辑元数据）。
4. 查看列出的对象，然后选择 Add metadata（添加元数据）。
5. 对于元数据 Type (类型)，请选择 System-defined (系统定义)。
6. 指定唯一的 Key (键) 和元数据 Value (值)。
7. 要编辑其他元数据，请选择 Add metadata (添加元数据)。您还可以选择 Remove（删除）以删除一组类型-键-值。
8. 完成后，请选择编辑元数据，Amazon S3 将编辑指定对象的元数据。

编辑用户定义的元数据

您可以通过组合元数据前缀 `x-amz-meta-` 和选择用于创建自定义键的名称来编辑对象的用户定义的元数据。例如，如果您添加自定义名称 `alt-name`，则元数据键为 `x-amz-meta-alt-name`。

用户定义元数据最大总计可为 2 KB。要计算用户定义元数据的总大小，请将 UTF-8 编码中每个键和值的字节数求和。键及其值均必须符合 US-ASCII 标准。有关更多信息，请参阅 用户定义的对象元数据。

编辑对象的用户定义的元数据

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择包含您想要将元数据添加到的对象的存储桶的名称。

   您还可以选择导航到文件夹。
3. 在 Objects（对象）列表中，选中要向其添加元数据的对象名称旁边的复选框。
4. 在 Actions（操作）菜单上，请选择 Edit metadata（编辑元数据）。
5. 查看列出的对象，然后选择 Add metadata（添加元数据）。
6. 对于元数据类型，请选择 User-defined（用户定义）。
7. 在 `x-amz-meta-` 后面输入唯一的自定义 Key（键）。还输入元数据 Value (值)。
8. 要添加其他元数据，请选择 Add metadata (添加元数据)。您还可以选择 Remove（删除）以删除一组类型-键-值。
9. 请选择 Edit metadata（编辑元数据）。

   Amazon S3 会编辑指定对象的元数据。

上传对象

在您将文件上传至 Amazon S3 时，文件会存储为 S3 对象。对象由文件数据和描述对象的元数据组成。一个存储桶中可以有无限量的对象。您需要拥有存储桶写入权限，才能将文件上传至 Amazon S3 存储桶。有关访问权限的更多信息，请参阅Amazon S3 的身份和访问管理。

您可以将任何类型的文件上传至 S3 存储桶，包括图像、备份、数据、电影等。可使用 Amazon S3 控制台上传的文件的最大大小为 160 GB。要上传大于 160GB 的文件，请使用 AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API。

如果启用了版本控制的存储桶中已存在所上传对象的键名，则 Amazon S3 会创建该对象的另一个版本，而不是替换现有对象。有关启用版本控制的更多信息，请参阅在存储桶上启用版本控制。

根据您要上传的数据大小，Amazon S3 提供以下选项：

- 使用 AWS SDK、REST API 或 AWS CLI 在单个操作中上传对象 – 您可以在单个 PUT 操作中上传最大为 5GB 的单个对象。
- 使用 Amazon S3 控制台上传单个对象 – 通过 Amazon S3 控制台，您可以上传最大 160GB 的单个对象。

- 使用 AWS SDK、REST API 或 AWS CLI 分段上传对象 – 您可以使用分段上传 API 操作来上传单个大型对象（最大大小为 5TB）。

  分段上传 API 操作旨在改进大型对象的上传体验。您可以分段上传对象。这些对象分段可以按任何顺序并行独立上传。您可以对大小在 5 MB 到 5 TB 范围内的对象使用分段上传。有关更多信息，请参阅 使用分段上传来上传和复制对象。

当您上传对象时，默认情况下，将使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）自动加密对象。当您下载对象时，该对象将被解密。有关更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为 和利用加密来保护数据。

上传对象时，如果您想使用不同类型的默认加密，也可以在 S3 PUT 请求中指定具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS），或者将目标存储桶中的默认加密配置设置为使用 SSE-KMS 来加密数据。有关 SSE-KMS 的更多信息，请参阅 使用 AWS KMS (SSE-KMS) 指定服务器端加密。如果您希望使用其他账户拥有的 KMS 密钥，则您必须有权使用该密钥。有关 KMS 密钥的跨账户权限的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建其他账户可以使用的 KMS 密钥。

如果您在 Amazon S3 中遇到拒绝访问（403 禁止访问）错误，请参阅排查 Amazon S3 中的拒绝访问（403 Forbidden）错误详细了解其常见原因。

上传对象

使用 S3 控制台

此过程介绍如何使用控制台将对象和文件夹上传到 Amazon S3 存储桶。

上传对象时，对象键名称是文件名和任何可选前缀。在 Amazon S3 控制台中，您可以创建文件夹来组织对象。在 Amazon S3 中，文件夹表示为出现在对象键名称中的前缀。如果您将单个对象上传到 Amazon S3 控制台中的文件夹，则文件夹名称将包含在对象键名称中。

例如，如果您将名为的对象上传 `sample1.jpg` 到名为 `backup` 的文件夹，则密钥名称为 `backup/sample1.jpg`。但是，对象会像 `sample1.jpg` 在 `backup` 文件夹中一样在控制台中显示。有关键名称的更多信息，请参阅 使用对象元数据。

> **Note**
>
> 如果在 Amazon S3 控制台中重命名对象或更改诸如存储类、加密或元数据等任何属性，将创建一个新对象来替换旧对象。如果启用 S3 版本控制，则会创建对象的新版本，而现有对象将变为旧版本。更改属性的角色也会成为新对象（或对象版本）的拥有者。

上传文件夹时，Amazon S3 会将所有文件和子文件夹从指定文件夹中上传至存储桶。然后，它会分配由上传文件名和文件夹名组成的对象键名。例如，如果您上传包含 `/images` 和 `sample1.jpg` 这两个文件的名为 `sample2.jpg` 的文件夹，则 Amazon S3 会上传这两个文件，然后分配相应的键名 `images/sample1.jpg` 和 `images/sample2.jpg`。键名包括作为前缀的文件夹名。Amazon S3 控制台仅显示最后一个“/”后面的键名部分。例如，在 `images` 文件夹中，`images/sample1.jpg` 和 `images/sample2.jpg` 对象显示为 `sample1.jpg` 和 `sample2.jpg`。

将文件夹和文件上传到 S3 存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（存储桶）列表中，请选择要将文件夹和文件上传到的存储桶的名称。
4. 请选择 Upload（上传）。
5. 在上传窗口中，执行下列操作之一：
   - 将文件和文件夹拖放到上传窗口。
   - 选择添加文件或添加文件夹，选择要上传的文件或文件夹，然后选择打开。
6. 要启用版本控制，请在目标下选择启用存储桶版本控制。
7. 要上传列出的文件和文件夹而不配置其他上传选项，请选择页面底部的上传。

   Amazon S3 会上传您的对象和文件夹。上传完成后，您可以在上传：状态页面上看到成功消息。

配置其他对象属性

1. 要更改访问控制列表权限，请选择 Permissions（权限）。
2. 在 Access control list (ACL) [访问控制列表 (ACL)]下，编辑权限。

   有关对象访问权限的信息，请参阅 使用 S3 控制台为对象设置 ACL 权限。您可以向公众（世界上的每一个人）授予对您的对象的读取访问权限，使其能够获取您正在上传的所有文件。但是，我们

建议您不要更改公共读取访问的默认设置。授予公有读取访问权限适用于一小部分的使用案例（如存储桶用于网站时）。您始终可以在上传对象后更改对象权限。

3. 要配置其他附加属性，请选择 Properties（属性）。
4. 在存储类下，为正在上传的文件选择存储类。

   有关存储类的更多信息，请参阅 使用 Amazon S3 存储类。
5. 要更新对象的加密设置，请在服务器端加密设置下执行以下操作：

   a. 选择 Specify an encryption key（指定加密密钥）。

   b. 在加密设置下，选择使用默认加密的存储桶设置或覆盖默认加密的存储桶设置。

   c. 如果您选择覆盖默认加密的存储桶设置，则必须配置以下加密设置。

   - 要使用由 Amazon S3 托管的密钥加密上传的文件，请选择 Amazon S3 托管式密钥（SSE-S3）。

     有关更多信息，请参阅 使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。
   - 要使用存储在 AWS Key Management Service（AWS KMS）中的密钥加密上传的文件，请选择 AWS Key Management Service 密钥（SSE-KMS）。然后，为 AWS KMS 密钥选择以下选项之一：
     - 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 中进行选择，然后从可用密钥的列表中选择您的 KMS 密钥。

       AWS 托管式密钥（aws/s3）和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。
     - 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
     - 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

       有关创建 AWS KMS key 的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

     > **Important**
     >
     > 您只能使用与存储桶所在相同的 AWS 区域中可用的 KMS 密钥。Amazon S3 控制台仅列出与存储桶位于同一区域中的前 100 个 KMS 密钥。要使用未列出的 KMS

密钥，您必须输入 KMS 密钥 ARN。如果您希望使用其他账户拥有的 KMS 密钥，则必须首先有权使用该密钥，然后必须输入相应的 KMS 密钥 ARN。
Amazon S3 仅支持对称加密 KMS 密钥，不支持非对称 KMS 密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的确定对称和非对称 KMS 密钥。

6. 要使用其他校验和，请选择 On（打开）。然后对于 Checksum function（校验和函数），选择要使用的函数。Amazon S3 在收到整个对象后计算并存储校验和值。您可以使用 Precalculated value（预计算值）框来提供预先计算的值。这样，Amazon S3 会将您提供的值与其计算的值进行比较。如果两个值不匹配，Amazon S3 将生成错误。

   其他校验和使您能够指定要用于验证数据的校验和算法。有关其他校验和的更多信息，请参阅检查对象完整性。

7. 要向上传的所有对象添加标签，请选择 Add tag (添加标签)。在键字段中输入标签名称。输入标签的值。

   对象标签为您提供了对存储进行分类的方法。每个标签都是一个键-值对。键和标签值区分大小写。对于每个对象，您最多可以有 10 个标签。标签键的长度最大可以为 128 个 Unicode 字符，标签值的长度最大可以为 255 个 Unicode 字符。有关对象标签的更多信息，请参阅 使用标签对存储进行分类。

8. 要添加元数据，请选择添加元数据。

   a. 在类型下，请选择系统定义或用户定义。

      对于系统定义元数据，您可以选择通用的 HTTP 标头，如 Content-Type 和 Content-Disposition。有关系统定义元数据的列表以及您能否添加值的信息，请参阅系统定义的对象元数据。以前缀 `x-amz-meta-` 开头的任何元数据都被视为用户定义的元数据。用户定义元数据会与对象存储在一起，并会在您下载该对象时返回。密钥及其值均必须符合 US-ASCII 标准。用户定义元数据最大可为 2 KB。如需详细了解系统定义和用户定义的元数据，请参阅使用对象元数据。

   b. 对于键，请选择一个键。

   c. 键入该键的值。

9. 要上传对象，请选择上传。

   Amazon S3 将上传您的对象。上传完成后，您可以在上传：状态页面上看到成功消息。

10. 请选择 Exit（退出）。

使用 AWS CLI

您可以发送 PUT 请求，在单个操作中上传最大 5 GB 的对象。有关更多信息，请参阅 AWS CLI 命令参考中的 `PutObject` 示例。

使用 REST API

您可以发送 REST 请求以上传对象。您可以发送 PUT 请求以在单个操作中上传数据。有关更多信息，请参阅 PUT Object。

使用 AWS SDK

您可以使用 AWS SDK 在 Amazon S3 中上传对象。SDK 向您提供了包装程序库，使您可以更轻松地上传数据。有关信息，请参阅支持的 SDK 列表。

以下是几个特色级 SDK 的示例：

.NET

以下 C# 代码示例将使用两个 `PutObjectRequest` 请求创建两个对象：

- 第一个 `PutObjectRequest` 请求将文本字符串保存为示例对象数据。它还指定存储桶和对象键名。
- 第二个 `PutObjectRequest` 请求通过指定文件名上传文件。此请求还指定 `ContentType` 标头和可选对象元数据（标题）。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadObjectTest
    {
        private const string bucketName = "*** bucket name ***";
        // For simplicity the example creates two objects from the same file.
        // You specify key names for these objects.
```

```
        private const string keyName1 = "*** key name for first object created ***";
        private const string keyName2 = "*** key name for second object created
***";
        private const string filePath = @"*** file path ***";
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.EUWest1;

        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                // 1. Put object-specify only key name for the new object.
                var putRequest1 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName1,
                    ContentBody = "sample text"
                };

                PutObjectResponse response1 = await
client.PutObjectAsync(putRequest1);

                // 2. Put the object-set ContentType and add metadata.
                var putRequest2 = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName2,
                    FilePath = filePath,
                    ContentType = "text/plain"
                };

                putRequest2.Metadata.Add("x-amz-meta-title", "someTitle");
                PutObjectResponse response2 = await
client.PutObjectAsync(putRequest2);
            }
            catch (AmazonS3Exception e)
```

```
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an
 object"
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Unknown encountered on server. Message:'{0}' when writing an
 object"
                    , e.Message);
            }
        }
    }
}
```

Java

以下示例将创建两个对象。第一个对象将一个文本字符串作为数据，第二对象是一个文件。该示例通过在对 AmazonS3Client.putObject() 的调用中直接指定存储桶名称、对象键和文本数据来创建第一个对象。该示例通过使用指定存储桶、对象键和文件路径的 PutObjectRequest 来创建第二个对象。PutObjectRequest 还指定 ContentType 标头和标题元数据。

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;
import java.io.IOException;

public class UploadObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
```

```
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Upload a text string as a new object.
            s3Client.putObject(bucketName, stringObjKeyName, "Uploaded String
 Object");

            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(bucketName,
 fileObjKeyName, new File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("title", "someTitle");
            request.setMetadata(metadata);
            s3Client.putObject(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

JavaScript

以下示例将现有文件上传到特定区域中的 Amazon S3 存储桶。

```
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});
```

```
export const main = async () => {
  const command = new PutObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
    Body: "Hello S3!",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

PHP

此示例将指导您使用 AWS SDK for PHP 中的类上传大小最大为 5GB 的对象。对于更大的文件，您必须使用分段上传 API 操作。有关更多信息，请参阅 使用分段上传来上传和复制对象。

有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

Example — 通过上传数据在 Amazon S3 存储桶中创建对象

通过使用 `putObject()` 方法来上传数据，以下 PHP 示例在指定存储桶中创建对象。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

try {
    // Upload data.
    $result = $s3->putObject([
```

```
        'Bucket' => $bucket,
        'Key'    => $keyname,
        'Body'   => 'Hello, world!',
        'ACL'    => 'public-read'
    ]);

    // Print the URL to the object.
    echo $result['ObjectURL'] . PHP_EOL;
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

AWS SDK for Ruby – 版本 3 通过两种方式将对象上传到 Amazon S3。第一种方式使用托管式文件上传程序，从而能更轻松地从磁盘上传任何大小的文件。使用托管文件上传程序方法：

1. 创建 `Aws::S3::Resource` 类的实例。
2. 按存储桶名称和键引用目标对象。位于存储桶中的对象具有可识别每个对象的唯一密钥。
3. 在对象上调用 `#upload_file`。

Example

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
```

```
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS SDK for Ruby - 版本 3 上传对象的第二种方式是使用 `Aws::S3::Object` 的 `#put` 方法。如果对象为字符串或者为不是磁盘上的文件的 I/O 对象，此方式很有用。要使用此方法，请执行以下操作：

1. 创建 `Aws::S3::Resource` 类的实例。
2. 按存储桶名称和键引用目标对象。
3. 调用 `#put`，从而传入字符串或 I/O 对象。

Example

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
```

```
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

在上传对象时使用有条件写入

在对上传操作使用有条件写入来创建对象之前，您可以检查存储桶中是否存在该对象。这样可以防止覆盖现有数据。有条件写入将验证存储桶中尚不存在具有相同键名称的现有对象。

可以通过在 UploadObject API 请求中指定值为 * 的可选 If-None-Match 条件标头来执行有条件写入。

有关有条件请求的更多信息，请参阅有条件请求。

使用分段上传来上传和复制对象

分段上传允许将单个对象作为一组分段上传。每个分段都是对象数据的连续部分。您可以独立上传以及按任意顺序上传这些对象分段。如果任意分段传输失败，可以重新传输该分段且不会影响其他分段。上传完所有的对象分段后，Amazon S3 将汇集这些分段并创建对象。一般而言，如果您的对象大小达到了 100 MB，您应该考虑使用分段上传，而不是在单个操作中上传对象。

使用分段上传可提供以下优势：

- 提高吞吐量 – 您可以并行上传分段以提高吞吐量。
- 从任何网络问题中快速恢复 – 较小的分段大小可以将由于网络错误而需重启失败的上传所产生的影响降至最低。
- 暂停和恢复对象上传 – 您可以在一段时间内逐步上传对象分段。启动分段上传后，不存在过期期限；您必须显式地完成或停止分段上传。
- 在您知道对象的最终大小前开始上传 – 您可以在创建对象时将其上传。

我们建议您按以下方式使用分段上传：

- 如果您在稳定的高带宽网络上传大型对象，请通过并行分段上传对象实现多线程上传，使用分段上传以充分利用您的可用带宽。
- 如果您在断点网络中上传对象，请使用分段上传以提高应对网络错误的复原能力，从而避免重新上传。在使用分段上传时，您只需重新尝试上传在上传期间中断的部分即可。而无需从头重新开始上传对象。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。有关将分段上传用于 S3 Express One Zone 和目录存储桶的更多信息，请参阅对目录桶使用分段上传。

分段上传流程

分段上分为三个步骤：开始上传、上传对象分段，以及在上传所有分段后完成分段上传。在收到完成分段上传请求后，Amazon S3 会利用上传的分段创建对象，然后您可以像在您的存储桶中访问任何其他对象一样访问该对象。

您可以列出所有正在执行的分段上传，或者获取为特定分段上传操作上传的分段列表。以上每个操作都在本节中进行了说明。

分段上传开始

当您发送请求以开始分段上传时，Amazon S3 将返回具有上传 ID 的响应，此 ID 是分段上传的唯一标识符。无论您何时上传分段、列出分段、完成上传或停止上传，您都必须包括此上传 ID。如果您想要提供描述已上传的对象的任何元数据，必须在请求中提供它以开始分段上传。

分段上传

上传分段时，除了指定上传 ID，还必须指定分段编号。您可以选择 1 和 10000 之间的任意分段编号。分段编号在您正在上传的对象中唯一地识别分段及其位置。您选择的分段编号不必是连续序列（例如，它可以是 1、5 和 14）。如果您使用之前上传的分段的同一分段编号上传新分段，则之前上传的分段将被覆盖。

当您上传某个分段时，Amazon S3 将在响应中返回此分段的实体标签（ETag）作为标头。对于每个分段上传，您必须记录分段编号和 ETag 值。您必须在随后的请求中包括这些值以完成分段上传。每个分段在上传时都有自己的 ETag。但是，一旦分段上传完成并且合并了所有分段，所有分段都将在一个 ETag 之下，作为多个校验和的校验和。

> **Note**
>
> 启动分段上传并上传一个或多个段之后，您必须完成或停止分段上传，才能不再被收取上传的分段的存储费用。只有在完成或停止分段上传之后，Amazon S3 才会释放分段存储并停止向您收取分段存储费用。
>
> 停止分段上传后，无法再次使用该上传 ID 上传任何分段。如果有任何分段上传正在进行，则即使在您停止上传后，它们仍然可能会成功或失败。为了确保释放所有分段使用的所有存储，必须仅在完成所有分段的上传后才停止分段上传。

分段上传完成

完成分段上传时，Amazon S3 通过按升序的分段编号规范化分段来创建对象。如果在开始分段上传请求中提供了任何对象元数据，则 Amazon S3 会将该元数据与对象相关联。成功完成请求后，分段将不再存在。

完成分段上传请求必须包括上传 ID 以及分段编号和相应的 ETag 值的列表。Amazon S3 响应包括可唯一地识别组合对象数据的 ETag。此 ETag 无需成为对象数据的 MD5 哈希。

分段上传调用示例

对于此示例，假设您正在为一个 100 GB 的文件生成分段上传。在这种情况下，您应在整个过程中进行以下 API 调用。总共将有 1002 个 API 调用。

- 一个用于启动该过程的 `CreateMultipartUpload` 调用。
- 1000 个单独的 `UploadPart` 调用，每次上传 100 MB 的一部分，总大小为 100 GB。
- 一个用于完成该过程的 `CompleteMultipartUpload` 调用。

分段上传列表

您可以列出特定分段上传或所有正在进行的分段上传的分段。列出分段操作将返回您已为特定分段上传而上传的分段信息。对于每个列出分段请求，Amazon S3 将返回有关特定分段上传的分段信息，最多为 1000 个分段。如果分段上传中的分段超过 1000 个，您必须发送一系列列出分段请求以检索所有分段。请注意，返回的分段列表不包括未完成上传的分段。使用列出分段上传 操作，您可以获得正在进行的分段上传的列表。

正在进行的分段上传是已开始但还未完成或停止的上传。每个请求将返回最多 1000 个分段上传。如果正在进行的分段上传超过 1000 个，您必须发送其他请求才能检索剩余的分段上传。仅使用返回的列表进行验证。发送完成分段上传 请求时，请勿使用此列表的结果。相反，当上传分段和 Amazon S3 返回的相应 ETag 值时，请保留您自己的指定分段编号的列表。

使用分段上传操作的校验和

在将对象上传到 Amazon S3 时，可指定校验和算法以供 Amazon S3 使用。默认情况下，Amazon S3 使用 MD5 来验证数据完整性；但是，您可以指定要使用的其他校验和算法。使用 MD5 时，Amazon S3 会在上传完成后计算整个分段对象的校验和。此校验和不是整个对象的校验和，而是每个分段的校验和的校验和。

当您指示 Amazon S3 使用其他校验和时，Amazon S3 会计算每个分段的校验和值并存储这些值。您可以使用 API 或 SDK 通过 `GetObject` 或 `HeadObject` 来检索单个分段的校验和值。如果您想检索仍在进行的分段上传的各个分段的校验和值，可以使用 `ListParts`。

> **Important**
>
> 如果您结合其他校验和使用分段上传，则分段编号必须是连续的编号。使用其他校验和时，如果您尝试使用非连续分段编号完成分段上传请求，Amazon S3 将生成 HTTP `500 Internal Server Error` 错误。

有关校验和如何处理分段对象的更多信息，请参阅检查对象完整性。

并发分段上传操作

在分布式开发环境中，您的应用程序可以同时在同一对象上开始多个更新。您的应用程序可能会使用同一对象键开始多个分段上传。然后，对于其中每个上传，您的应用程序可以上传分段并将完成上传请求发送到 Amazon S3，以创建对象。当存储桶启用了 S3 版本控制时，完成分段上传将始终创建一个新版本。当您在启用版本控制的存储桶中启动使用相同对象键的多个分段上传时，对象的当前版本将由最新（`createdDate`）开始的上传决定。例如，假设您在上午 10:00 启动对某个对象的 CreateMultipartUpload 请求。然后，您在上午 11:00 提交对同一对象的第二个 `CreateMultipartUpload` 请求。因为第二个请求是最新提交的，所以上午 11:00 的请求所上传的对象将是当前版本，即使第一个上传是在第二个上传之后完成的，也是如此。对于未启用版本控制的存储桶，在开始分段上传和完成分段上传期间接收的某些其他请求可能会优先开始。

> **Note**
>
> 在开始分段上传和完成分段上传期间接收的某些其他请求可能会优先开始。例如，如果在使用键开始分段上传之后，但在完成分段上传之前其他操作删除了该键，则完成分段上传响应可能表示在未看到对象的情况下即成功创建了对象。

在分段上传中使用有条件写入

在对上传操作使用有条件写入来创建对象之前，您可以检查存储桶中是否存在该对象。这样可以防止覆盖现有数据。有条件写入将验证存储桶中尚不存在具有相同键名称的现有对象。

可以通过在 `CompleteMultipartUpload` API 请求中指定值为 `*` 的可选 `If-None-Match` 条件标头来执行有条件写入。

有关有条件请求的更多信息，请参阅有条件请求。

分段上传和定价

开始分段上传后，Amazon S3 将保留所有分段，直到您完成或停止上传。在整个其生命周期内，您将支付有关此分段上传及其关联分段的所有存储、带宽和请求的费用。

这些分段根据上传分段时指定的存储类收费。这种情况的一个例外是上传到 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 的分段。在上传完成之前，针对 S3 Glacier Flexible Retrieval 存储类的 PUT 请求的正在上传的分段采用 S3 Standard 存储费率按 S3 Glacier Flexible Retrieval 暂存存储计费。此外，CreateMultipartUpload 和 UploadPart 均按 S3 Standard 费率计费。只有 CompleteMultipartUpload 请求按照 S3 Glacier Flexible Retrieval 费率计费。同样，在上传完成之前，

针对 S3 Glacier Deep Archive 存储类的 PUT 请求的正在上传的分段采用 S3 Standard 存储费率按 S3 Glacier Flexible Retrieval 暂存存储计费，而只有 CompleteMultipartUpload 请求按 S3 Glacier Deep Archive 费率计费。

如果您停止分段上传，Amazon S3 将删除上传构件和已上传的任何分段，您将不再支付它们的费用。无论指定的存储类如何，删除未完成的分段上传均不收取提前删除费用。有关定价的更多信息，请参阅 Amazon S3 定价。

> **Note**
>
> 为了最大程度地降低存储成本，我们建议您配置生命周期规则，以便使用 `AbortIncompleteMultipartUpload` 操作在指定的天数后删除未完成的分段上传。有关创建生命周期规则以删除未完成的分段上传的更多信息，请参阅配置存储桶生命周期配置以删除未完成的分段上传。

分段上传的 API 支持

这些库提供了易于上传分段对象的高级别抽象。但是，如果需要您的应用程序，您可以直接使用 REST API。Amazon Simple Storage Service API 参考的下面几节描述了适用于分段上传的 REST API。

有关使用 AWS Lambda 函数的分段上传演练，请参阅 Uploading large objects to Amazon S3 using multipart upload and transfer acceleration。

- 创建分段上传
- 上传分段
- 上传分段（复制）
- 完成分段上传
- 中止分段上传
- 列出分段
- 列出分段上传

AWS Command Line Interface 对于分段上传的支持

AWS Command Line Interface 中的以下主题介绍了适用于分段上传的操作。

- 开始分段上传

- 上传分段
- 上传分段（复制）
- 完成分段上传
- 中止分段上传
- 列出分段
- 列出分段上传

AWS SDK 对于分段上传的支持

您可以使用 AWS SDK 分段上传对象。有关 API 操作支持的 AWS SDK 的列表，请参阅：

- 创建分段上传
- 上传分段
- 上传分段（复制）
- 完成分段上传
- 中止分段上传
- 列出分段
- 列出分段上传

分段上传 API 和权限

您必须具有使用分段上传操作的所需权限。您可以使用访问控制列表（ACL）、存储桶策略或用户策略来授予个人执行这些操作的权限。下表列出了使用 ACL、存储桶策略或用户策略时，各种分段上传操作的所需权限。

操作	所需权限
创建分段上传	必须允许您对对象执行 `s3:PutObject` 操作，才能开始分段上传。 存储桶拥有者可以允许其他主体执行 `s3:PutObject` 操作。
开始分段上传	必须允许您对对象执行 `s3:PutObject` 操作，才能开始分段上传。 存储桶拥有者可以允许其他主体执行 `s3:PutObject` 操作。

操作	所需权限
发起者	标识分段上传发起者的容器元素。如果发起者是 AWS 账户，此元素将提供与 Owner 元素相同的信息。如果发起者是 IAM 用户，此元素将提供用户 ARN 和显示名称。
上传分段	必须允许您对对象执行 s3:PutObject 操作，才能上传分段。 存储桶拥有者必须允许发起者对对象执行 s3:PutObject 操作，以便发起者可以上传该对象的分段。
上传分段（复制）	必须允许您对对象执行 s3:PutObject 操作，才能上传分段。因为您正在上传现有对象的分段，因此必须允许您对源对象执行 s3:GetObject 。 存储桶拥有者必须允许发起者对对象执行 s3:PutObject 操作，发起者才能上传该对象的分段。
完成分段上传	必须允许您对对象执行 s3:PutObject 操作，才能完成分段上传。 存储桶拥有者必须允许发起者对对象执行 s3:PutObject 操作，以便发起者可以完成该对象的分段上传。
停止分段上传	必须允许您执行 s3:AbortMultipartUpload 操作，才能停止分段上传。 默认情况下，允许存储桶拥有者和分段上传的发起者执行此操作，作为 IAM 和存储桶策略的一部分。如果发起者是 IAM 用户，也允许该用户的 AWS 账户停止此分段上传。使用 VPC 端点策略，分段上传的发起者不会自动获得执行 s3:AbortMultipartUpload 操作的权限。 除了这些默认情况之外，存储桶拥有者可以允许其他主体对对象执行 s3:AbortMultipartUpload 操作。存储桶拥有者可以拒绝任何主体，使其无法执行 s3:AbortMultipartUpload 操作。

操作	所需权限
列出分段	您必须得到可以执行 s3:ListMultipartUploadParts 操作的允许，才能在分段上传中列出分段。 在默认情况下，存储桶拥有者有权为任何针对存储桶的分段上传列出分段。分段上传的发起者有权为特定分段上传列出分段。如果分段上传的发起者是 IAM 用户，则控制该 IAM 用户的 AWS 账户 同样有权列出此次上传的分段。 除了这些默认情况之外，存储桶拥有者可以允许其他主体对对象执行 s3:ListMultipartUploadParts 操作。存储桶拥有者也可以拒绝任何主体，使其无法执行 s3:ListMultipartUploadParts 操作。
列出分段上传	您必须得到可以对存储桶执行 s3:ListBucketMultipartUploads 操作的允许，才能列出正在上传到该存储桶的分段上传。 除了默认情况之外，存储桶拥有者可以允许其他主体对存储桶执行 s3:ListBucketMultipartUploads 操作。
AWS KMS 加密和解密相关权限	要使用具有 AWS Key Management Service (AWS KMS) KMS 密钥的加密执行分段上传，请求者必须有权对密钥执行 kms:Decrypt 和 kms:GenerateDataKey 操作。请求者还必须拥有对 CreateMultipartUpload API 执行 kms:GenerateDataKey 操作的权限。然后，请求者需要对 UploadPart 和 UploadPartCopy API 执行 kms:Decrypt 操作的权限。这些权限是必需的，因为 Amazon S3 必须在完成分段上传之前解密并读取加密的文件段中的数据。 如果您的 IAM 用户或角色位于与 KMS 密钥相同的 AWS 账户 中，您必须在密钥策略中具有这些权限。如果您的 IAM 用户或角色属于与 KMS 密钥不同的账户，您必须在密钥策略和 IAM 用户或角色中具有这些权限。

有关 ACL 权限与访问策略中的权限之间关系的信息，请参阅 ACL 权限和访问策略权限的映射。有关 IAM 用户、角色和最佳实践的信息，请参阅《IAM 用户指南》中的 IAM 身份（用户、用户组和角色）。

主题

- 配置存储桶生命周期配置以删除未完成的分段上传
- 使用分段上传操作上传对象

- 使用高级别 .NET TransferUtility 类上传目录。
- 列出分段上传
- 跟踪分段上传
- 中止分段上传
- 使用分段上传复制对象
- Amazon S3 分段上传限制

配置存储桶生命周期配置以删除未完成的分段上传

作为最佳实践，我们建议您使用 AbortIncompleteMultipartUpload 操作配置生命周期规则，以最大程度地降低存储成本。有关中止分段上传的更多信息，请参阅 中止分段上传。

Amazon S3 支持存储桶生命周期规则，您可以使用该规则指示 Amazon S3 停止未在启动后的指定天数内完成的分段上传。如果分段上传未在指定的时间范围内完成，则符合中止操作的条件。然后，Amazon S3 停止分段上传，并删除与分段上传关联的分段。此规则既适用于现有的分段上传，也适用于您后续创建的分段上传。

下面是使用 AbortIncompleteMultipartUpload 操作指定规则的示例生命周期配置。

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Prefix></Prefix>
        <Status>Enabled</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

在本示例中，该规则不为 Prefix 元素（对象键名称前缀）指定值。因此，该规则适用于您启动了分段上传的存储桶中的所有对象。任何已启动但未在七天内完成的分段上传都符合执行中止操作的条件。中止操作对已完成的分段上传没有任何效果。

有关存储桶生命周期配置的更多信息，请参阅 管理存储生命周期。

> **Note**
>
> 如果分段上传在规则中指定的天数内完成，则 `AbortIncompleteMultipartUpload` 生命周期操作不适用（即，Amazon S3 不会执行任何操作）。此外，此操作不适用于对象。此生命周期操作不会删除任何对象。此外，当您删除任何未完成的分段上传分段时，不会收取 S3 生命周期的提前删除费用。

使用 S3 控制台

要自动管理未完成的分段上传，您可以使用 S3 控制台创建生命周期规则，以使存储桶中的未完成分段上传字节在指定天数后过期。以下过程介绍如何添加生命周期规则，以在 7 天后删除未完成的分段上传。有关添加生命周期规则的更多信息，请参阅在存储桶上设置生命周期配置。

添加生命周期规则以中止超过 7 天的未完成的分段上传

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要为其创建生命周期规则的存储桶的名称。
3. 请选择 Management（管理）选项卡，然后选择 Create lifecycle rule（创建生命周期规则）。
4. 在 Lifecycle rule name (生命周期规则名称) 中，输入规则的名称。

   在该存储桶内，此名称必须是唯一的。
5. 请选择生命周期规则的范围：

   - 要为具有特定前缀的所有对象创建生命周期规则，请选择 Limit the scope of this rule using one or more filters（使用一个或多个筛选条件限制此规则的范围），然后在 Prefix（前缀）字段中输入前缀。
   - 要为存储桶中的所有对象创建生命周期规则，请选择 This rule applies to all objects in the bucket（此规则适用于存储桶中的所有对象），然后选择 I acknowledge that this rule applies to all objects in the bucket（我确认此规则适用于存储桶中的所有对象）。
6. 在 Lifecycle rule actions（生命周期规则操作）下，选择 Delete expired object delete markers or incomplete multipart uploads（删除过期的对象删除标记或未完成的分段上传）。
7. 在 Delete expired object delete markers or incomplete multipart uploads（删除过期的对象删除标记或未完成的分段上传）下，选择 Delete incomplete multipart uploads（删除未完成的分段上传）。

8. 在 Number of days（天数）字段中，输入天数，在此天数之后将删除未完成的分段上传（在本例中为 7 天）。
9. 选择创建规则。

使用 AWS CLI

以下 `put-bucket-lifecycle-configuration` AWS Command Line Interface（AWS CLI）命令为指定的存储桶添加生命周期配置。要使用此命令，请将 *user input placeholders* 替换为您的信息。

```
aws s3api put-bucket-lifecycle-configuration  \
        --bucket amzn-s3-demo-bucket1  \
        --lifecycle-configuration filename-containing-lifecycle-configuration
```

以下示例显示如何使用 AWS CLI 添加生命周期规则以中止未完成的分段上传。它包括一个示例 JSON 生命周期配置，以中止超过 7 天的未完成的分段上传。

要使用本示例中的 CLI 命令，请将 *user input placeholders* 替换为您的信息。

添加生命周期规则以中止未完成的分段上传

1. 设置 AWS CLI。有关说明，请参阅 使用 AWS CLI 进行 Amazon S3 开发。
2. 将以下示例生命周期配置保存在一个文件（例如 *lifecycle.json*）中。该示例配置指定了空前缀，因此它适用于存储桶中的所有对象。为将配置限制为某个对象子集，您可以指定前缀。

```
{
    "Rules": [
        {
            "ID": "Test Rule",
            "Status": "Enabled",
            "Filter": {
                "Prefix": ""
            },
            "AbortIncompleteMultipartUpload": {
                "DaysAfterInitiation": 7
            }
        }
    ]
}
```

3. 运行以下 CLI 命令以在存储桶上设置此生命周期配置。

```
aws s3api put-bucket-lifecycle-configuration   \
--bucket amzn-s3-demo-bucket1  \
--lifecycle-configuration file://lifecycle.json
```

4. 要验证是否已在您的存储桶上设置了生命周期配置，请使用以下 `get-bucket-lifecycle` 命令检索生命周期配置。

```
aws s3api get-bucket-lifecycle  \
--bucket amzn-s3-demo-bucket1
```

5. 要删除生命周期配置，请使用以下 `delete-bucket-lifecycle` 命令。

```
aws s3api delete-bucket-lifecycle \
--bucket amzn-s3-demo-bucket1
```

使用分段上传操作上传对象

您可以使用分段上传以编程方式将单个对象上传到 Amazon S3。

有关更多信息，请参阅以下部分。

使用 S3 控制台

您可以将任何类型的文件上传至 S3 存储桶，包括图像、备份、数据、电影等。可使用 Amazon S3 控制台上传的文件的最大大小为 160 GB。要上传大于 160GB 的文件，请使用 AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API。

有关通过 AWS Management Console上传对象的说明，请参阅上传对象。

使用 AWS CLI

AWS Command Line Interface (AWS CLI) 中的以下各部分介绍了适用于分段上传的操作。

- 开始分段上传
- 上传分段
- 上传分段（复制）
- 完成分段上传
- 中止分段上传
- 列出分段

- 列出分段上传

您也可以使用 REST API 创建您自己的 REST 请求，或者也可以使用其中一个 AWS SDK。有关 REST API 的更多信息，请参阅使用 REST API。有关 SDK 的更多信息，请参阅 使用分段上传操作上传对象。

使用 REST API

Amazon Simple Storage Service API 参考的下面几节描述了适用于分段上传的 REST API。

- 开始分段上传
- 上传分段
- 完成分段上传
- 停止分段上传
- 列出分段
- 列出分段上传

使用 AWS SDK

有关如何使用 AWS SDK 执行分段上传的示例，请参阅使用 AWS SDK 执行 Amazon S3 对象的分段上传。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

使用 AWS SDK（高级别 API）

一些 AWS SDK 公开了一个高级别 API，该 API 通过将完成分段上传所需的不同 API 操作组合成单个操作，来简化分段上传。有关更多信息，请参阅 使用分段上传来上传和复制对象。

如果您需要暂停并恢复分段上传、在上传期间改变分段大小，或者事先不知道数据大小，请使用低级别 API 方法。用于分段上传的低级别 API 方法提供了额外功能，有关更多信息，请参阅使用 AWS SDK（低级别 API）。

Java

有关如何使用 Java SDK 执行分段上传的示例，请参阅使用 AWS SDK 执行 Amazon S3 对象的分段上传。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

.NET

要将文件上传到 S3 存储桶，请使用 `TransferUtility` 类。在从文件上传数据时，您必须提供对象的键名。如果未提供，该 API 将使用文件名作为键名。在从流上传数据时，您必须提供对象的键名。

要设置高级上传选项（如段大小、并发上传段时的线程数、元数据、存储类或 ACL），请使用 `TransferUtilityUploadRequest` 类。

> **Note**
>
> 如果您将流用作数据源，`TransferUtility` 类不会执行并发上传。

以下 C# 示例将文件分段上传到 Amazon S3 存储桶。它说明如何使用各种 `TransferUtility.Upload` 重载来上传文件。每个对上传的后续调用都将替换先前的上传。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPUHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object
 ***";
        private const string filePath = "*** provide the full path name of the file
 to upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
```

```
            s3Client = new AmazonS3Client(bucketRegion);
            UploadFileAsync().Wait();
        }

        private static async Task UploadFileAsync()
        {
            try
            {
                var fileTransferUtility =
                    new TransferUtility(s3Client);

                // Option 1. Upload a file. The file name is used as the object key
name.
                await fileTransferUtility.UploadAsync(filePath, bucketName);
                Console.WriteLine("Upload 1 completed");

                // Option 2. Specify object key name explicitly.
                await fileTransferUtility.UploadAsync(filePath, bucketName,
keyName);
                Console.WriteLine("Upload 2 completed");

                // Option 3. Upload data from a type of System.IO.Stream.
                using (var fileToUpload =
                    new FileStream(filePath, FileMode.Open, FileAccess.Read))
                {
                    await fileTransferUtility.UploadAsync(fileToUpload,
                                               bucketName, keyName);
                }
                Console.WriteLine("Upload 3 completed");

                // Option 4. Specify advanced settings.
                var fileTransferUtilityRequest = new TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    FilePath = filePath,
                    StorageClass = S3StorageClass.StandardInfrequentAccess,
                    PartSize = 6291456, // 6 MB.
                    Key = keyName,
                    CannedACL = S3CannedACL.PublicRead
                };
                fileTransferUtilityRequest.Metadata.Add("param1", "Value1");
                fileTransferUtilityRequest.Metadata.Add("param2", "Value2");

                await fileTransferUtility.UploadAsync(fileTransferUtilityRequest);
```

```
                Console.WriteLine("Upload 4 completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }

        }
    }
}
```

JavaScript

Example

上传大文件。

```
import {
  CreateMultipartUploadCommand,
  UploadPartCommand,
  CompleteMultipartUploadCommand,
  AbortMultipartUploadCommand,
  S3Client,
} from "@aws-sdk/client-s3";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

export const main = async () => {
  const s3Client = new S3Client({});
  const bucketName = "test-bucket";
  const key = "multipart.txt";
  const str = createString();
  const buffer = Buffer.from(str, "utf8");
```

```
let uploadId;

try {
  const multipartUpload = await s3Client.send(
    new CreateMultipartUploadCommand({
      Bucket: bucketName,
      Key: key,
    }),
  );

  uploadId = multipartUpload.UploadId;

  const uploadPromises = [];
  // Multipart uploads require a minimum size of 5 MB per part.
  const partSize = Math.ceil(buffer.length / 5);

  // Upload each part.
  for (let i = 0; i < 5; i++) {
    const start = i * partSize;
    const end = start + partSize;
    uploadPromises.push(
      s3Client
        .send(
          new UploadPartCommand({
            Bucket: bucketName,
            Key: key,
            UploadId: uploadId,
            Body: buffer.subarray(start, end),
            PartNumber: i + 1,
          }),
        )
        .then((d) => {
          console.log("Part", i + 1, "uploaded");
          return d;
        }),
    );
  }

  const uploadResults = await Promise.all(uploadPromises);

  return await s3Client.send(
    new CompleteMultipartUploadCommand({
      Bucket: bucketName,
      Key: key,
```

```
        UploadId: uploadId,
        MultipartUpload: {
          Parts: uploadResults.map(({ ETag }, i) => ({
            ETag,
            PartNumber: i + 1,
          })),
        },
      }),
    );

    // Verify the output by downloading the file from the Amazon Simple Storage
 Service (Amazon S3) console.
    // Because the output is a 25 MB string, text editors might struggle to open the
 file.
  } catch (err) {
    console.error(err);

    if (uploadId) {
      const abortCommand = new AbortMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
      });

      await s3Client.send(abortCommand);
    }
  }
};
```

Example

下载大文件。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream } from "fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
```

```
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: parseInt(start),
    end: parseInt(end),
    length: parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

// When downloading a large file, you might want to break it down into
// smaller pieces. Amazon S3 accepts a Range header to specify the start
// and end of the byte range to be downloaded.
const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    console.log(`Downloading bytes ${nextRange.start} to ${nextRange.end}`);

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
```

```
  }
};

export const main = async () => {
  await downloadInChunks({
    bucket: "my-cool-bucket",
    key: "my-cool-object.txt",
  });
};
```

Go

Example

使用上传管理器上传大型对象，以将数据分成多个分段，然后同时上传。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}
```

```
// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
```

```
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
}
```

Example

使用下载管理器下载大型对象，以分段获取数据并同时下载它们。

```
// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey string)
 ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

PHP

本主题介绍如何使用 Aws\S3\Model\MultipartUpload\UploadBuilder 中的高级别 AWS SDK for PHP 类执行文件分段上传。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

以下 PHP 代码示例将文件上传到 Amazon S3 存储桶。该示例演示如何设置 `MultipartUploader` 对象的参数。

```
 require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Prepare the upload parameters.
$uploader = new MultipartUploader($s3, '/path/to/large/file.zip', [
    'bucket' => $bucket,
    'key'    => $keyname
]);

// Perform the upload.
try {
    $result = $uploader->upload();
    echo "Upload complete: {$result['ObjectURL']}" . PHP_EOL;
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Python

以下示例使用高级别分段上传 Python API（`TransferManager` 类）上传对象。

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig
```

```
MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
```

```
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
```

```
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
```

```
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
```

```
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

使用 AWS SDK（低级别 API）

AWS SDK 公开了一个与 Amazon S3 REST API 非常相似的用于分段上传的低级别 API（请参阅使用分段上传来上传和复制对象）。当您需要暂停和恢复分段上传、在上传过程中更改部分大小或事先不知道上传数据的大小时，请使用低级别 API。当您没有这些需求时，请使用高级别 API（请参阅使用 AWS SDK（高级别 API））。

Java

以下示例演示如何使用低级别 Java 类上传文件。它将执行以下步骤：

- 使用 `AmazonS3Client.initiateMultipartUpload()` 方法初始化分段上传，并传入 `InitiateMultipartUploadRequest` 对象。
- 保存 `AmazonS3Client.initiateMultipartUpload()` 方法返回的上传 ID。您为随后的每个分段上传操作提供此上传 ID。
- 上传对象的分段。对于每个分段，您将调用 `AmazonS3Client.uploadPart()` 方法。您使用 `UploadPartRequest` 对象提供分段上传信息。

- 对于每个分段，在列表中保存来自 AmazonS3Client.uploadPart() 方法的响应的 ETag。您使用 ETag 值完成分段上传。
- 调用 AmazonS3Client.completeMultipartUpload() 方法来完成分段上传。

Example

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartUpload {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String filePath = "*** Path to file to upload ***";

        File file = new File(filePath);
        long contentLength = file.length();
        long partSize = 5 * 1024 * 1024; // Set part size to 5 MB.

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Create a list of ETag objects. You retrieve ETags for each object
 part
```

```
            // uploaded,
            // then, after each individual part has been uploaded, pass the list of
ETags to
            // the request to complete the upload.
            List<PartETag> partETags = new ArrayList<PartETag>();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest(bucketName, keyName);
            InitiateMultipartUploadResult initResponse =
s3Client.initiateMultipartUpload(initRequest);

            // Upload the file parts.
            long filePosition = 0;
            for (int i = 1; filePosition < contentLength; i++) {
                // Because the last part could be less than 5 MB, adjust the part
size as
                // needed.
                partSize = Math.min(partSize, (contentLength - filePosition));

                // Create the request to upload a part.
                UploadPartRequest uploadRequest = new UploadPartRequest()
                        .withBucketName(bucketName)
                        .withKey(keyName)
                        .withUploadId(initResponse.getUploadId())
                        .withPartNumber(i)
                        .withFileOffset(filePosition)
                        .withFile(file)
                        .withPartSize(partSize);

                // Upload the part and add the response's ETag to our list.
                UploadPartResult uploadResult = s3Client.uploadPart(uploadRequest);
                partETags.add(uploadResult.getPartETag());

                filePosition += partSize;
            }

            // Complete the multipart upload.
            CompleteMultipartUploadRequest compRequest = new
CompleteMultipartUploadRequest(bucketName, keyName,
                    initResponse.getUploadId(), partETags);
            s3Client.completeMultipartUpload(compRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
```

```
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

以下 C# 示例演示如何使用低级别 AWS SDK for .NET 分段上传 API 将文件上传到 S3 存储桶。有关 Amazon S3 分段上传的信息，请参阅 使用分段上传来上传和复制对象。

> **Note**
>
> 如果使用 AWS SDK for .NET API 上传大型对象，数据写入到请求流时可能出现超时。您可以使用 `UploadPartRequest` 设置显式超时。

以下 C# 示例使用低级别分段上传 API 将文件上传到 S3 存储桶。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPULowLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object
 ***";
```

```
        private const string filePath = "*** provide the full path name of the file
to upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Uploading an object");
            UploadObjectAsync().Wait();
        }

        private static async Task UploadObjectAsync()
        {
            // Create list to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = bucketName,
                Key = keyName
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Upload parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                Console.WriteLine("Uploading parts");

                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
```

```
                        {
                            BucketName = bucketName,
                            Key = keyName,
                            UploadId = initResponse.UploadId,
                            PartNumber = i,
                            PartSize = partSize,
                            FilePosition = filePosition,
                            FilePath = filePath
                        };

                    // Track upload progress.
                    uploadRequest.StreamTransferProgress +=
                        new
EventHandler<StreamTransferProgressArgs>(UploadPartProgressEventCallback);

                    // Upload a part and add the response to our list.
                    uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Setup to complete the upload.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        UploadId = initResponse.UploadId
                     };
                completeRequest.AddPartETags(uploadResponses);

                // Complete the upload.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown: { 0}",
exception.Message);

                // Abort the upload.
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
```

```
                {
                    BucketName = bucketName,
                    Key = keyName,
                    UploadId = initResponse.UploadId
                };
               await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
        public static void UploadPartProgressEventCallback(object sender,
 StreamTransferProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

PHP

本主题说明如何使用 AWS SDK for PHP 版本 3 中的低级别 uploadPart 方法来以分段形式上传文件。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

以下 PHP 示例使用低级别 PHP API 分段上传将文件上传到 Amazon S3 存储桶。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$filename = '*** Path to and Name of the File to Upload ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

$result = $s3->createMultipartUpload([
    'Bucket'       => $bucket,
    'Key'          => $keyname,
    'StorageClass' => 'REDUCED_REDUNDANCY',
```

```
    'Metadata'     => [
        'param1' => 'value 1',
        'param2' => 'value 2',
        'param3' => 'value 3'
    ]
]);
$uploadId = $result['UploadId'];

// Upload the file in parts.
try {
    $file = fopen($filename, 'r');
    $partNumber = 1;
    while (!feof($file)) {
        $result = $s3->uploadPart([
            'Bucket'     => $bucket,
            'Key'        => $keyname,
            'UploadId'   => $uploadId,
            'PartNumber' => $partNumber,
            'Body'       => fread($file, 5 * 1024 * 1024),
        ]);
        $parts['Parts'][$partNumber] = [
            'PartNumber' => $partNumber,
            'ETag' => $result['ETag'],
        ];
        $partNumber++;

        echo "Uploading part $partNumber of $filename." . PHP_EOL;
    }
    fclose($file);
} catch (S3Exception $e) {
    $result = $s3->abortMultipartUpload([
        'Bucket'   => $bucket,
        'Key'      => $keyname,
        'UploadId' => $uploadId
    ]);

    echo "Upload of $filename failed." . PHP_EOL;
}

// Complete the multipart upload.
$result = $s3->completeMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
```

```
    'MultipartUpload'    => $parts,
]);
$url = $result['Location'];

echo "Uploaded $filename to $url." . PHP_EOL;
```

使用 AWS SDK for Ruby

AWS SDK for Ruby 版本 3 支持两种 Amazon S3 分段上传方式。对于第一个选项，您可以使用托管文件上传。有关更多信息，请参阅 AWS 开发人员博客中的将文件上传至 Amazon S3。托管文件上传是建议用于将文件上传到存储桶的方法。它们提供以下优势：

- 管理大于 15MB 的对象的分段上传。
- 正确打开二进制模式的文件，规避编码问题。
- 在并行上传较大对象的多个部分时，使用多个线程。

此外，您还可以直接使用以下分段上传客户端操作：

- create_multipart_upload – 启动分段上传，并返回上传 ID。
- upload_part – 上传分段上传中的一部分。
- upload_part_copy – 将现有对象作为数据源，并复制其中的数据，上传一部分。
- complete_multipart_upload – 整合先前上传的部分，完成分段上传。
- abort_multipart_upload – 停止分段上传。

使用高级别 .NET TransferUtility 类上传目录。

您可以使用 `TransferUtility` 类上传整个目录。默认情况下，该 API 仅上传位于指定目录的根目录中的文件。但是，您可以指定以递归方式上传所有子目录中的文件。

要根据筛选条件选择指定目录中的文件，请指定筛选表达式。例如，要从目录中仅上传 .pdf 文件，请指定 `"*.pdf"` 筛选表达式。

在从目录中上传文件时，您不必为生成的对象指定键名。Amazon S3 会使用原始文件路径构造键名。例如，假设您有一个名为 `c:\myfolder` 的目录，并且此目录具有以下结构：

Example

```
C:\myfolder
```

```
      \a.txt
      \b.pdf
      \media\
             An.mp3
```

上传此目录时，Amazon S3 将使用以下密钥名称：

Example

```
a.txt
b.pdf
media/An.mp3
```

Example

以下 C# 代码示例将一个目录上传到 Amazon S3 存储桶。它说明如何使用各种 `TransferUtility.UploadDirectory` 重载来上传目录。每个对上传的后续调用都将替换先前的上传。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadDirMPUHighLevelAPITest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string directoryPath = @"*** directory path ***";
        // The example uploads only .txt files.
        private const string wildCard = "*.txt";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadDirAsync().Wait();
        }
```

```
        private static async Task UploadDirAsync()
        {
            try
            {
                var directoryTransferUtility =
                    new TransferUtility(s3Client);

                // 1. Upload a directory.
                await directoryTransferUtility.UploadDirectoryAsync(directoryPath,
                    existingBucketName);
                Console.WriteLine("Upload statement 1 completed");

                // 2. Upload only the .txt files from a directory
                //    and search recursively.
                await directoryTransferUtility.UploadDirectoryAsync(
                                                directoryPath,
                                                existingBucketName,
                                                wildCard,
                                                SearchOption.AllDirectories);
                Console.WriteLine("Upload statement 2 completed");

                // 3. The same as Step 2 and some optional configuration.
                //    Search recursively for .txt files to upload.
                var request = new TransferUtilityUploadDirectoryRequest
                {
                    BucketName = existingBucketName,
                    Directory = directoryPath,
                    SearchOption = SearchOption.AllDirectories,
                    SearchPattern = wildCard
                };

                await directoryTransferUtility.UploadDirectoryAsync(request);
                Console.WriteLine("Upload statement 3 completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an object",
e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
```

```
                    "Unknown encountered on server. Message:'{0}' when writing an
 object", e.Message);
            }
        }
    }
}
```

列出分段上传

您可以使用 AWS SDK（低级别 API）在 Amazon S3 中检索正在进行的分段上传的列表。

使用 AWS SDK（低级别 API）列出分段上传

Java

以下任务将引导您使用低级别 Java 类来列出存储桶上所有正在进行的分段上传。

低级别 API 分段上传列表过程

1	创建 `ListMultipartUploadsRequest` 类的实例并提供存储桶名称。
2	运行 `AmazonS3Client.listMultipartUploads` 方法。该方法将返回 `MultipartUploadListing` 类的实例，以向您提供有关正在进行的分段上传的信息。

以下 Java 代码示例演示了上述任务。

Example

```
ListMultipartUploadsRequest allMultpartUploadsRequest =
     new ListMultipartUploadsRequest(existingBucketName);
MultipartUploadListing multipartUploadListing =
     s3Client.listMultipartUploads(allMultpartUploadsRequest);
```

.NET

要列出特定存储桶上所有正在进行的分段上传，请使用 AWS SDK for .NET 低级别分段上传 API 的 `ListMultipartUploadsRequest` 类。`AmazonS3Client.ListMultipartUploads` 方法将返回 `ListMultipartUploadsResponse` 类（提供有关正在进行的分段上传的信息）的实例。

正在进行的分段上传是使用启动分段上传请求启动但尚未完成或停止的分段上传。有关 Amazon S3 分段上传的更多信息，请参阅 使用分段上传来上传和复制对象。

以下 C# 示例演示如何使用 AWS SDK for .NET 列出存储桶上所有正在进行的分段上传。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
ListMultipartUploadsRequest request = new ListMultipartUploadsRequest
{
  BucketName = bucketName // Bucket receiving the uploads.
};

ListMultipartUploadsResponse response = await
 AmazonS3Client.ListMultipartUploadsAsync(request);
```

PHP

本主题说明如何使用版本 3 的 AWS SDK for PHP 中的低级别 API 类列出存储桶上所有正在进行的分段上传。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

以下 PHP 示例演示如何列出存储桶上所有正在进行的分段上传。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Retrieve a list of the current multipart uploads.
$result = $s3->listMultipartUploads([
    'Bucket' => $bucket
]);

// Write the list of uploads to the page.
print_r($result->toArray());
```

使用 REST API 列出分段上传

Amazon Simple Storage Service API 参考的下面几节描述了列出分段上传的 REST API：

- ListParts- 列出特定分段上传的已上传部分。
- ListMultipartUploads- 列出正在进行的分段上传。

使用 AWS CLI 列出分段上传

AWS Command Line Interface 中的以下各部分介绍了适用于列出分段上传的操作。

- list-parts- 列出特定分段上传的已上传部分。
- list-multipart-uploads- 列出正在进行的分段上传。

跟踪分段上传

高级别分段上传 API 提供了侦听接口 `ProgressListener`，用于在将对象上传到 Amazon S3 时跟踪上传进度。进度事件将定期发生，并且会通知侦听者已传输的字节。

Java

Example

```
TransferManager tm = new TransferManager(new ProfileCredentialsProvider());

PutObjectRequest request = new PutObjectRequest(
    existingBucketName, keyName, new File(filePath));

// Subscribe to the event and provide event handler.
request.setProgressListener(new ProgressListener() {
   public void progressChanged(ProgressEvent event) {
    System.out.println("Transferred bytes: " +
      event.getBytesTransfered());
            }
});
```

Example

以下 Java 代码将上传文件并使用 `ProgressListener` 来跟踪上传进度。有关如何创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import java.io.File;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.event.ProgressEvent;
import com.amazonaws.event.ProgressListener;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.Upload;

public class TrackMPUProgressUsingHighLevelAPI {

    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide bucket name ***";
        String keyName            = "*** Provide object key ***";
        String filePath           = "*** file to upload ***";

        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());


        // For more advanced uploads, you can create a request object
        // and supply additional request parameters (ex: progress listeners,
        // canned ACLs, etc.)
        PutObjectRequest request = new PutObjectRequest(
          existingBucketName, keyName, new File(filePath));

        // You can ask the upload for its progress, or you can
        // add a ProgressListener to your request to receive notifications
        // when bytes are transferred.
        request.setGeneralProgressListener(new ProgressListener() {
   @Override
   public void progressChanged(ProgressEvent progressEvent) {
    System.out.println("Transferred bytes: " +
      progressEvent.getBytesTransferred());
   }
  });

        // TransferManager processes all transfers asynchronously,
        // so this call will return immediately.
        Upload upload = tm.upload(request);

        try {
         // You can block and wait for the upload to finish
```

```
            upload.waitForCompletion();
        } catch (AmazonClientException amazonClientException) {
            System.out.println("Unable to upload file, upload aborted.");
            amazonClientException.printStackTrace();
        }
    }
}
```

.NET

以下 C# 示例使用 TransferUtility 类将文件上传到 S3 存储桶并跟踪上传的进度。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TrackMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide the bucket name ***";
        private const string keyName = "*** provide the name for the uploaded object
 ***";
        private const string filePath = " *** provide the full path name of the file
 to upload **";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;


        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            TrackMPUAsync().Wait();
        }

        private static async Task TrackMPUAsync()
        {
```

```
            try
            {
                var fileTransferUtility = new TransferUtility(s3Client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>
                        (uploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }

        static void uploadRequest_UploadPartProgressEvent(object sender,
 UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

中止分段上传

启动分段上传后，您便开始上传段。Amazon S3 将存储这些段，但它只有在您上传所有这些段并发送完成分段上传的 successful 请求（您应验证您的完成分段上传的请求是否成功）之后，才会利用这些段创建对象。在收到完成分段上传请求后，Amazon S3 会将这些段汇集在一起并创建一个对象。如果您未成功发送完成分段上传请求，则 Amazon S3 不会汇集这些段并且不会创建任何对象。

您需要为与上传的分段关联的所有存储付费。有关更多信息，请参阅 分段上传和定价。因此，完成分段上传以创建对象，或者停止分段上传以删除任意已上传的分段非常重要。

您可以使用 AWS Command Line Interface（AWS CLI）、REST API 或 AWS SDK 停止 Amazon S3 中正在进行的分段上传。您还可以使用存储桶生命周期配置停止未完成的分段上传。

使用 AWS 软件开发工具包（高级别 API）

Java

TransferManager 类提供了停止正在进行的分段上传的 abortMultipartUploads 方法。启动上传后，上传将被视为正在进行，直到您完成或停止该操作。您可以提供 Date 值和此 API 来停止该存储桶上所有在指定 Date 前启动的并且仍在进行的上传。

以下任务将引导您使用高级别 Java 类来停止分段上传。

高级别 API 分段上传停止过程

1	创建 TransferManager 类的实例。
2	通过传递存储桶名称和 Date 值执行 TransferManager.abortMultipartUploads 方法。

以下 Java 代码将停止特定存储桶上所有在一周前启动的并且仍在进行的分段上传。有关如何创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import java.util.Date;

import com.amazonaws.AmazonClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.transfer.TransferManager;

public class AbortMPUUsingHighLevelAPI {
```

```
    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide existing bucket name ***";

        TransferManager tm = new TransferManager(new ProfileCredentialsProvider());


        int sevenDays = 1000 * 60 * 60 * 24 * 7;
  Date oneWeekAgo = new Date(System.currentTimeMillis() - sevenDays);

        try {
         tm.abortMultipartUploads(existingBucketName, oneWeekAgo);
        } catch (AmazonClientException amazonClientException) {
         System.out.println("Unable to upload file, upload was aborted.");
         amazonClientException.printStackTrace();
        }
    }
}
```

> **Note**
>
> 您也可以停止特定的分段上传。有关更多信息，请参阅 使用 AWS SDK（低级别 API）。

.NET

以下 C# 示例将停止一周前在特定存储桶上启动的正在进行的所有分段上传。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class AbortMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        // Specify your bucket region (an example region is shown).
```

```
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            AbortMPUAsync().Wait();
        }

        private static async Task AbortMPUAsync()
        {
            try
            {
                var transferUtility = new TransferUtility(s3Client);

                // Abort all in-progress uploads initiated before the specified
 date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

> **Note**
>
> 您也可以停止特定的分段上传。有关更多信息，请参阅 使用 AWS SDK（低级别 API）。

使用 AWS SDK（低级别 API）

您可以通过调用 AmazonS3.abortMultipartUpload 方法停止正在进行的分段上传。此方法将删除任何已上传到 Amazon S3 的分段并释放资源。您必须提供上传 ID、存储桶名称和键名称。以下 Java 代码示例演示如何停止正在进行的分段上传。

要停止分段上传，请提供上传时使用的上传 ID、存储桶名称和键名。停止一个分段上传之后，您便无法使用相应的上传 ID 上传其他分段。有关 Amazon S3 分段上传的更多信息，请参阅 使用分段上传来上传和复制对象。

Java

以下 Java 代码示例停止正在进行的分段上传。

Example

```
InitiateMultipartUploadRequest initRequest =
    new InitiateMultipartUploadRequest(existingBucketName, keyName);
InitiateMultipartUploadResult initResponse =
               s3Client.initiateMultipartUpload(initRequest);

AmazonS3 s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
s3Client.abortMultipartUpload(new AbortMultipartUploadRequest(
            existingBucketName, keyName, initResponse.getUploadId()));
```

> **Note**
>
> 您可以在特定时间之前停止所有仍在进行的分段上传，而不是停止某个特定的分段上传。此清理操作对于停止您已启动但未完成或停止的旧分段上传非常有用。有关更多信息，请参阅使用 AWS 软件开发工具包（高级别 API）。

.NET

以下 C# 示例演示如何停止分段上传。有关包含以下代码的完整 C# 示例，请参阅 使用 AWS SDK（低级别 API）。

```
AbortMultipartUploadRequest abortMPURequest = new AbortMultipartUploadRequest
{
    BucketName = existingBucketName,
```

```
    Key = keyName,
    UploadId = initResponse.UploadId
};
await AmazonS3Client.AbortMultipartUploadAsync(abortMPURequest);
```

还可以中止在特定时间之前启动的所有正在进行的分段上传。对于中止未完成或已中止的分段上传，此清理操作很有用。有关更多信息，请参阅 使用 AWS 软件开发工具包（高级别 API）。

PHP

本示例介绍如何使用 AWS SDK for PHP 的版本 3 中的类中止正在进行的分段上传。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。例如，`abortMultipartUpload()` 方法。

有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$uploadId = '*** Upload ID of upload to Abort ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Abort the multipart upload.
$s3->abortMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
]);
```

使用 REST API

有关使用 REST API 停止分段上传的更多信息，请参阅 Amazon Simple Storage Service API 参考中的 AbortMultipartUpload。

使用 AWS CLI

有关使用 AWS CLI 停止分段上传的更多信息，请参阅《AWS CLI 命令参考》中的 abort-multipart-upload。

使用分段上传复制对象

本节中的示例将向您展示如何使用分段上传 API 复制大于 5 GB 的对象。您可以在单个操作中复制小于 5 GB 的对象。有关更多信息，请参阅 复制、移动和重命名对象。

使用 AWS SDK

要使用低级别 API 复制对象，请执行以下操作：

- 通过调用 `AmazonS3Client.initiateMultipartUpload()` 方法启动分段上传。
- 从 `AmazonS3Client.initiateMultipartUpload()` 方法返回的响应对象保存上传 ID。您为每个分段上传操作提供此上传 ID。
- 复制所有段。对于需要复制的每个段，创建一个 `CopyPartRequest` 类的新实例。提供段信息，包括源和目标存储桶名称、源和目标对象键、上传 ID、段的第一个字节和最后一个字节的位置以及段编号。
- 保存 `AmazonS3Client.copyPart()` 方法调用的响应。每个响应均包括 `ETag` 值和已上传分段的段编号。您需要此信息才能完成分段上传。
- 调用 `AmazonS3Client.completeMultipartUpload()` 方法以完成复制操作。

Java

Example

以下示例说明如何使用 Amazon S3 低级别 Java API 执行分段复制。有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;
```

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartCopy {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String sourceBucketName = "*** Source bucket name ***";
        String sourceObjectKey = "*** Source object key ***";
        String destBucketName = "*** Target bucket name ***";
        String destObjectKey = "*** Target object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(destBucketName,
                    destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(sourceBucketName, sourceObjectKey);
            ObjectMetadata metadataResult =
 s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make
 sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
 1);
```

```
                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(sourceBucketName)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(destBucketName)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and
make the
            // copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    destBucketName,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
```

```
    }
}
```

.NET

以下 C# 示例说明如何使用 AWS SDK for .NET 将大于 5 GB 的 Amazon S3 对象从一个源位置复制到另一个源位置，例如从一个存储桶复制到另一个存储桶。要复制小于 5 GB 的对象，请使用 使用 AWS SDK 中所述的单个操作复制过程。有关 Amazon S3 分段上传的更多信息，请参阅 使用分段上传来上传和复制对象。

此示例说明如何使用 AWS SDK for .NET 分段上传 API 将大于 5 GB 的 Amazon S3 对象从一个 S3 存储桶复制到另一个存储桶。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectUsingMPUapiTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with
 source object ***";
        private const string targetBucket = "*** provide the name of the bucket to
 copy the object to ***";
        private const string sourceObjectKey = "*** provide the name of object to
 copy ***";
        private const string targetObjectKey = "*** provide the name of the object
 copy ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            MPUCopyObjectAsync().Wait();
```

```
        }
        private static async Task MPUCopyObjectAsync()
        {
            // Create a list to store the upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();
            List<CopyPartResponse> copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest =
                new InitiateMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey
                };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            String uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = sourceBucket,
                    Key = sourceObjectKey
                };

                GetObjectMetadataResponse metadataResponse =
                    await s3Client.GetObjectMetadataAsync(metadataRequest);
                long objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                long partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
```

```
                    CopyPartRequest copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = targetBucket,
                        DestinationKey = targetObjectKey,
                        SourceBucket = sourceBucket,
                        SourceKey = sourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i
                    };

                    copyResponses.Add(await s3Client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                CompleteMultipartUploadRequest completeRequest =
                new CompleteMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey,
                    UploadId = initResponse.UploadId
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }
    }
```

```
}
```

使用 REST API

Amazon Simple Storage Service API 参考的下面几节描述了适用于分段上传的 REST API。使用上传分段（复制）API 复制现有的对象，并通过在请求中添加 `x-amz-copy-source` 请求标头指定源对象。

- 开始分段上传
- 上传分段
- 上传分段（复制）
- 完成分段上传
- 中止分段上传
- 列出分段
- 列出分段上传

您可以使用这些 API 生成您自己的 REST 请求，也可以使用我们提供的 SDK 之一。有关通过 AWS CLI 执行分段上传的更多信息，请参阅 使用 AWS CLI。有关 SDK 的更多信息，请参阅 AWS SDK 对于分段上传的支持。

Amazon S3 分段上传限制

下表提供了分段上传的核心规范。有关更多信息，请参阅 使用分段上传来上传和复制对象。

Item	规范
最大对象大小	5 TiB
每次上传的分段的最大数量	10000
分段编号	1 到 10,000（含）
分段大小	5MiB 到 5GiB。未对分段上传的最后一段施加最小大小限制。
列出分段请求返回的分段的最大数量	1000

Item	规范
在列出分段上传请求中返回的分段的最大数量	1000

有条件请求

有条件请求支持您向 S3 操作添加前提条件。使用有条件请求时，您可以向 READ 和 WRITE 请求添加标头。这些标头指定了相关条件，如果不满足这些条件，将导致 S3 操作失败。

有条件请求无需额外付费。对于适用的请求，包括失败的请求，您只需按现有费率付费。有关 Amazon S3 特征和定价的信息，请参阅 Amazon S3 定价。

主题

有条件读取

通过有条件读取，您可以在读取请求中使用其它标头，以便向 S3 操作添加前提条件。如果不满足这些前提条件，请求将失败。

以下 S3 API 支持使用有条件读取：

- GetObject
- HeadObject
- CopyObject

可以使用以下标头来返回依赖于实体标签（ETag）或上次修改日期的对象。有关对象元数据（例如 ETag 和 Last-Modified）的更多信息，请参阅the section called “系统定义的对象元数据”。

GetObject

- If-Match — 仅当对象的 ETag 与提供的 ETag 匹配时，才返回该对象。

- If-Modified-Since — 仅当对象自指定时间以来已被修改时，才返回该对象。
- If-None-Match — 仅当对象的 ETag 与提供的 ETag 不匹配时，才返回该对象。
- If-Unmodified-Since — 仅当对象自指定时间以来未被修改时，才返回该对象。

有关这些标头、返回的错误以及 S3 在单个请求中处理多个条件标头的顺序的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 `GetObject`。

`HeadObject`

- If-Match — 仅当对象的 ETag 与提供的 ETag 匹配时，才返回该对象。
- If-Modified-Since — 仅当对象自指定时间以来已被修改时，才返回该对象。
- If-None-Match — 仅当对象的 ETag 与提供的 ETag 不匹配时，才返回该对象。
- If-Unmodified-Since — 仅当对象自指定时间以来未被修改时，才返回该对象。

有关这些标头、返回的错误以及 S3 在单个请求中处理多个条件标头的顺序的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 `HeadObject`。

`CopyObject`

- x-amz-copy-source-if-match — 仅当对象的 ETag 与提供的 ETag 匹配时，才复制源对象。
- x-amz-copy-source-if-modified-since — 仅当对象自指定时间以来已被修改时，才复制源对象。
- x-amz-copy-source-if-none-match — 仅当对象的 ETag 与提供的 ETag 不匹配时，才复制源对象。
- x-amz-copy-source-if-unmodified-since — 仅当对象自指定时间以来未被修改时，才复制源对象。

有关这些标头、返回的错误以及 S3 在单个请求中处理多个条件标头的顺序的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 `CopyObject`。

有条件写入

通过有条件写入，您可以在写入请求中使用其它标头，以便向 S3 操作添加前提条件。这样可以防止覆盖现有数据。有条件写入将验证存储桶中尚不存在具有相同键名称的现有对象。

要执行有条件写入，您必须拥有 `s3:PutObject` 权限。这使调用方能够检查存储桶中是否存在对象。可以在 AWS SDK 中使用带有预签名 URL 的有条件写入。

> **Note**
>
> 要使用有条件写入，必须通过 HTTPS（TLS）发出请求，或使用 AWS 签名版本 4 对请求进行签名。

以下 S3 API 支持使用有条件写入：

- PutObject
- CompleteMultipartUpload

可以使用以下标头来写入依赖于对象键名称的对象。有关对象键名称的一般信息，请参阅the section called “创建对象键”。

PutObject

- If-None-Match — 仅当指定存储桶中不存在具有相同键名称的现有对象时，才上传对象。需要 *（星号）值。

使用 AWS CLI

以下 `put-object` 示例命令说明了如何使用 AWS CLI，通过 `if-none-match` 参数借助有条件写入标头上传对象。

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key dir-1/my_images.tar.bz2 --body
 my_images.tar.bz2 --if-none-match "*"
```

有关更多信息，请参阅 AWS CLI 命令参考 中的 put-object。

有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的什么是 AWS Command Line Interface？。

有关此标头的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutObject。

CompleteMultipartUpload

- If-None-Match — 仅当指定存储桶中不存在具有相同键名称的现有对象时，才完成上传。需要 *（星号）值。

使用 AWS CLI

以下 complete-multipart-upload 示例命令说明了如何使用 AWS CLI，通过 if-none-match 参数借助有条件写入标头完成分段上传。

```
aws s3api complete-multipart-upload --multipart-upload file://mpustruct --bucket amzn-s3-demo-bucket --key dir-1/my_images.tar.bz2 --upload-id uploadID --if-none-match "*"
```

有关更多信息，请参阅 AWS CLI 命令参考 中的 complete-multipart-upload。

有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的什么是 AWS Command Line Interface？。

有关此标头的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CompleteMultipartUpload。

有条件写入行为

有条件写入会针对存储桶中的现有对象进行评估。如果存储桶中不存在具有相同键名称的现有对象，则写入操作将成功并导致 200 响应。如果存在现有对象，则写入操作将失败并导致 412 Precondition Failed 响应。对于启用了版本控制的存储桶，作为有条件评估的一部分，S3 会检查是否存在同名的当前对象版本。如果当前没有同名的对象版本，或者当前对象版本是删除标记，则写入操作将成功。否则，它会导致写入操作失败和 412 Precondition Failed 响应。

如果对同一个对象名称进行多个有条件写入，则第一个要完成的写入操作将成功。然后，Amazon S3 使后续写入失败并生成 412 Precondition Failed 响应。

如果在针对对象的有条件写入操作完成之前，对于对象的删除请求获得成功，则在并发请求的情况下也可能收到 409 Conflict 响应。这是因为删除请求优先于更早启动的有条件写入操作。将有条件写入与 PutObject 结合使用时，可能会在收到 409 错误后重试上传。使用 CompleteMultipartUpload 时，必须使用 CreateMultipartUpload 重新启动分段上传，以便在收到 409 错误后再次上传对象。

考虑以下场景，即两个客户端对同一个存储桶运行操作。

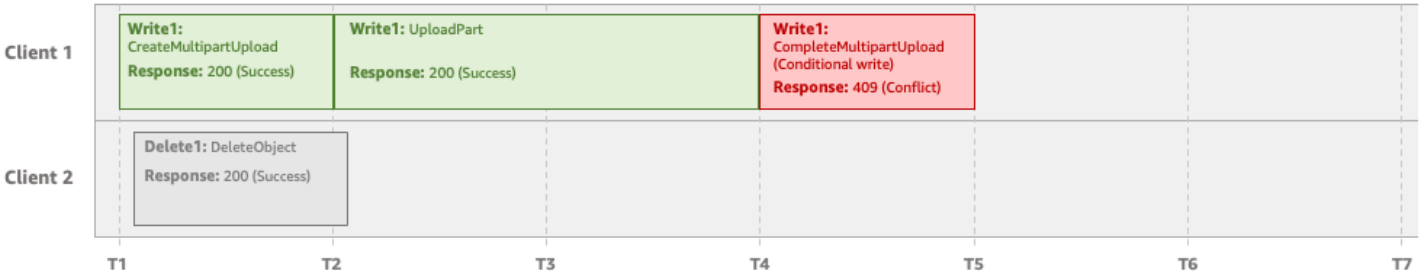
412 前提条件失败响应

有条件写入不考虑任何正在进行的分段上传请求，因为这些对象还不是完全写入的对象。请考虑以下示例，其中客户端 1 正在使用分段上传来上传对象。在分段上传期间，客户端 2 能够通过有条件写入操

作成功写入相同的对象。随后，当客户端 1 尝试使用有条件写入完成分段上传时，上传将失败，因为对象已经存在。

409 冲突响应

如果在有条件写入请求可以完成之前，删除请求获得成功，则 Amazon S3 为写入操作返回 409 Conflict 响应。这是因为更早启动的删除请求优先于有条件写入操作。考虑以下示例，其中存储桶中存在文件 puppy.txt。客户端 1 开始对另一个同样名为 puppy.txt 的文件进行分段上传，目的是通过有条件写入来完成分段上传。在上传过程中，客户端 2 从存储桶中删除 puppy.txt。当客户端 1 尝试将 CompleteMultipartUpload 和有条件写入结合使用时，它将失败并导致 409 Conflict 响应。在这种情况下，您必须启动新的分段上传。



> **Note**
>
> 为了最大程度地降低存储成本，我们建议您配置生命周期规则，以便使用 AbortIncompleteMultipartUpload 操作在指定的天数后删除未完成的分段上传。有关创建生命周期规则以删除未完成的分段上传的更多信息，请参阅配置存储桶生命周期配置以删除未完成的分段上传。

复制、移动和重命名对象

CopyObject 操作将创建已存储在 Amazon S3 中的对象的副本。

在单个原子操作中，您可以创建最大 5GB 的对象副本。但是，要复制大于 5GB 的对象，您必须使用分段上传。有关更多信息，请参阅 the section called “复制对象”。

通过使用 CopyObject 操作，您可以：

- 创建对象的其它副本。
- 通过复制对象并删除原始对象来重命名它们。
- 将对象从一个存储桶复制或移动到另一个存储桶，包括跨 AWS 区域（例如，从 us-west-1 到 eu-west-2）。移动对象时，Amazon S3 将对象复制到指定目标，然后删除源对象。

  > **Note**
  >
  > 跨 AWS 区域复制或移动对象会产生带宽费用。有关更多信息，请参阅 Amazon S3 定价。

- 更改对象元数据。每个 Amazon S3 对象都有元数据。此元数据是一组名称/值对。您可以在上传对象时设置对象元数据。上传对象后，您将无法修改对象元数据。修改对象元数据的唯一方式是创建对象的副本并设置元数据。为此，在复制操作中，将相同的对象设置为源和目标。

  有些对象元数据是系统元数据，而另外一些则是用户定义的元数据。您可以控制某些系统元数据。例如，您可以控制要用于对象的存储类和服务器端加密的类型。复制对象时，还会复制用户控制的系统元数据和用户定义的元数据。Amazon S3 将重置系统控制的元数据。例如，在复制对象时，Amazon S3 将重置已复制对象的创建日期。您不需要在复制请求中设置任何这些系统控制的元数据值。

  复制对象时，您可能会决定更新某些元数据值。例如，如果您的源对象被配置为使用 S3 Standard 存储，您可以选择对对象复制使用 S3 Intelligent-Tiering。您可能还会决定更改源对象上某些用户定义的元数据值。请注意，如果您选择在复制期间更新任意对象的用户可配置元数据（系统或用户定义的元数据），则必须显式指定请求中源对象上存在的所有用户可配置的元数据，即使您只更改其中一个元数据值也是如此。

  有关对象元数据的详细信息，请参阅 使用对象元数据。

复制已存档和已还原的对象

如果源对象归档在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 中，您必须先还原临时副本，然后才能将对象复制到另一个存储桶中。有关对象归档的更多信息，请参阅 转换为 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类（对象存档）。

在 Amazon S3 控制台中，不支持对 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中的已还原对象执行复制操作。要复制这些已还原对象，请使用 AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API。

复制加密对象

Amazon S3 会自动加密所有复制到 S3 存储桶的新对象。如果您未在复制请求中指定加密信息，则目标对象的加密设置将设为目标存储桶的默认加密配置。默认情况下，所有存储桶都有基本级别的加密配置，该配置使用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）。如果目标存储桶的默认加密配置使用具有 AWS Key Management Service（AWS KMS）密钥（SSE-KMS）或客户提供加密密钥（SSE-C）的服务器端加密，则 Amazon S3 使用相应的 KMS 密钥或客户提供的密钥来加密目标对象副本。

在复制对象时，如果您希望对目标对象使用其他类型的加密设置，您可以请求 Amazon S3 使用 KMS 密钥、Amazon S3 托管式密钥或客户提供的密钥对目标对象加密。如果您的请求中的加密设置与目标存储桶的默认加密配置不同，则您的请求中的加密设置优先。如果副本的源对象是用 SSE-C 加密的，您必须在请求中提供必需的加密信息，以便 Amazon S3 可以解密对象以进行复制。有关更多信息，请参阅 利用加密来保护数据。

复制对象时使用校验和

复制对象时，可以选择对对象使用不同的校验和算法。无论您选择使用相同的算法还是新算法，Amazon S3 都会在复制对象后计算一个新的校验和值。Amazon S3 不会直接复制校验和的值。通过使用分段上传加载的对象的校验和值可能会发生变化。有关如何计算此校验和的更多信息，请参阅使用分段级别墅校验和进行分段上传。

在单个请求中复制多个对象

要使用单个请求复制多个 Amazon S3 对象，您还可以使用 S3 批量操作。您为 S3 批量操作提供要操作的对象列表。S3 批量操作调用相应的 API 操作来执行指定的操作。单个批量操作任务可对包含 EB 级数据的数十亿个对象执行指定操作。

S3 批量操作特征包括跟踪进度、发送通知并存储所有操作的详细完成报告，从而提供完全托管、可审核的无服务器体验。您可以通过 Amazon S3 控制台、AWS CLI、AWS SDK 或 REST API 使用 S3 批量操作。有关更多信息，请参阅 the section called “批量操作基础知识”。

将对象复制到目录存储桶

有关将对象复制到目录存储桶的信息，请参阅将对象复制到目录存储桶。有关将 Amazon S3 Express One Zone 存储类与目录存储桶结合使用的信息，请参阅什么是 S3 Express One Zone？和目录桶。

复制对象

要复制对象，请使用以下方法。

使用 S3 控制台

> **Note**
>
> - 使用 Amazon S3 控制台复制对象时，您必须具有 `s3:ListAllMyBuckets` 权限。控制台需要此权限以验证复制操作。有关授予此权限的示例策略，请参阅the section called “基于身份的策略示例”。
>
> 如果您要复制具有用户定义标签的对象，您还必须拥有 `s3:GetObjectTagging` 权限。如果您要复制的对象没有用户定义的标签，但大小超过 16 MB，您还必须拥有 `s3:GetObjectTagging` 权限。
>
> 如果目标存储桶策略拒绝 `s3:GetObjectTagging` 操作，则会复制对象，但不包含用户定义的标签，并且您将收到错误。
> - 无法使用 S3 控制台复制使用客户提供的加密密钥（SSE-C）加密的对象。要复制使用 SSE-C 加密的对象，请使用 AWS CLI、AWS SDK 或 Amazon S3 REST API。
> - Amazon S3 控制台不支持跨区域复制使用 SSE-KMS 加密的对象。要跨区域复制使用 SSE-KMS 加密的对象，请使用 AWS CLI、AWS SDK 或 Amazon S3 REST API。

复制对象

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶，然后选择通用存储桶选项卡。导航到包含待复制对象的 Amazon S3 存储桶或文件夹。
3. 选中要复制的对象名称左侧的复选框。
4. 在操作菜单上，从显示的选项列表中选择复制。
5. 选择目标类型和目标账户。要指定目标路径，请选择 Browse S3（浏览 S3），导航到目标，然后选中目标左侧的复选框。选择右下角的选择目标。

   或者，输入目标路径。

6. 如果未启用存储桶版本控制，则系统可能会要求您确认是否覆盖具有相同名称的现有对象。如果可以覆盖，请选中该复选框并继续。如果要在此存储桶中保留对象的所有版本，请选择 Enable Bucket Versioning（启用存储桶版本控制）。您还可以更新默认加密和 S3 对象锁定属性。
7. 在 Additional checksums（其他校验和）下，选择是要使用现有校验和函数复制对象，还是用新校验和函数替换现有校验和函数。上传对象时，您可以选择指定用于验证数据完整性的校验和算法。复制对象时，您可以选择新函数。如果您最初没有指定额外的校验和，则可以使用复制选项的这一部分添加一个校验和。

   > **Note**
   >
   > 即使您选择使用相同的校验和函数，如果您复制对象且对象大小超过 16 MB，校验和值也可能会发生变化。由于分段上传的校验和计算方式的原因，校验和值可能会发生变化。有关在复制对象时校验和可能会如何变化的信息，请参阅使用分段级别墅校验和进行分段上传。

   要更改校验和函数，请选择用新的校验和函数替换。从框中选择新的校验和函数。复制对象时，将使用指定的算法计算和存储新校验和。
8. 选择右下角的复制。Amazon S3 会将对象复制到目标。

使用 AWS SDK

本节中的示例向您展示了如何复制单个操作中大于 5 GB 的对象。要复制大于 5GB 的对象，您必须使用分段上传。有关更多信息，请参阅 使用分段上传复制对象。

Java

Example

以下示例使用 AWS SDK for Java 复制 Amazon S3 中的对象。有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

```
import com.amazonaws.services.s3.model.CopyObjectRequest;

import java.io.IOException;

public class CopyObjectSingleOperation {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String sourceKey = "*** Source object key *** ";
        String destinationKey = "*** Destination object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjRequest = new CopyObjectRequest(bucketName,
 sourceKey, bucketName, destinationKey);
            s3Client.copyObject(copyObjRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

以下 C# 示例使用高级别 AWS SDK for .NET 在单次操作中复制最大为 5 GB 的对象。对于大于 5 GB 的对象，请使用 使用分段上传复制对象 中所述的分段上传复制示例。

此示例将创建最大为 5 GB 的对象的副本。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
```

```
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with
 source object ***";
        private const string destinationBucket = "*** provide the name of the bucket
 to copy the object to ***";
        private const string objectKey = "*** provide the name of object to copy
 ***";
        private const string destObjectKey = "*** provide the destination object key
 name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            CopyingObjectAsync().Wait();
        }

        private static async Task CopyingObjectAsync()
        {
            try
            {
                CopyObjectRequest request = new CopyObjectRequest
                {
                    SourceBucket = sourceBucket,
                    SourceKey = objectKey,
                    DestinationBucket = destinationBucket,
                    DestinationKey = destObjectKey
                };
                CopyObjectResponse response = await
 s3Client.CopyObjectAsync(request);
            }
            catch (AmazonS3Exception e)
```

```
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

此主题将指导您使用第 3 版 AWS SDK for PHP，将 Amazon S3 中的单个对象和多个对象从一个存储桶复制到另一个存储桶或者在同一存储桶中进行复制。

有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

以下 PHP 示例演示使用 `copyObject()` 方法复制 Amazon S3 中的单个对象。它还演示如何通过使用 `getcommand()` 方法对 `CopyObject` 进行批量调用来制作对象的多个副本。

复制对象

1	使用 `Aws\S3\S3Client` 类构造函数创建 Amazon S3 客户端的实例。
2	要制作对象的多个副本，请运行对 Amazon S3 客户端 getCommand() 方法的批量调用，该方法是从 Aws\CommandInterface 类继承的。您提供 `CopyObject` 命令作为第一个参数，提供包含源存储桶、源键名、目标存储桶和目标键名的数组作为第二个参数。

```
 require 'vendor/autoload.php';

use Aws\CommandPool;
use Aws\Exception\AwsException;
use Aws\ResultInterface;
use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
```

```
$sourceKeyname = '*** Your Source Object Key ***';
$targetBucket = '*** Your Target Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Copy an object.
$s3->copyObject([
    'Bucket' => $targetBucket,
    'Key' => "$sourceKeyname-copy",
    'CopySource' => "$sourceBucket/$sourceKeyname",
]);

// Perform a batch of CopyObject operations.
$batch = array();
for ($i = 1; $i <= 3; $i++) {
    $batch[] = $s3->getCommand('CopyObject', [
        'Bucket' => $targetBucket,
        'Key' => "{targetKeyname}-$i",
        'CopySource' => "$sourceBucket/$sourceKeyname",
    ]);
}
try {
    $results = CommandPool::batch($s3, $batch);
    foreach ($results as $result) {
        if ($result instanceof ResultInterface) {
            // Result handling here
        }
        if ($result instanceof AwsException) {
            // AwsException handling here
        }
    }
} catch (Exception $e) {
    // General error handling here
}
```

Python

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
```

```
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key
```

```
    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

Ruby

以下任务将引导您使用 Ruby 类将 Amazon S3 中的对象从一个存储桶复制到另一个存储桶，或者在同一个存储桶内复制。

复制对象

1	对 AWS SDK for Ruby 的版本 3 使用 Amazon S3 模块化 Gem 时，需要 `aws-sdk-s3` 并提供您的 AWS 凭证。有关如何提供您的凭证的更多信息，请参阅 使用 AWS 账户 或 IAM 用户凭证发出请求。
2	提供源存储桶名称、源键名、目标存储桶名称和目标键等请求信息。

下面的 Ruby 代码示例通过使用 `#copy_object` 方法将对象从一个存储桶复制到另一个存储桶，演示了上述任务。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
```

```
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 REST API

本示例描述了如何使用 Amazon S3 REST API 复制对象。有关 REST API 的更多信息，请参阅CopyObject。

本示例将 flotsam 存储桶中的 *amzn-s3-demo-bucket1* 对象复制到 jetsam 存储桶的 *amzn-s3-demo-bucket2* 对象，同时保留其元数据。

```
PUT /jetsam HTTP/1.1
Host: amzn-s3-demo-bucket2.s3.amazonaws.com
x-amz-copy-source: /amzn-s3-demo-bucket1/flotsam
Authorization: AWS AKIAIOSFODNN7EXAMPLE:ENoSbxYByFA0UGLZUqJN5EUnLDg=
Date: Wed, 20 Feb 2008 22:12:21 +0000
```

将从以下信息生成签名。

```
PUT\r\n
```

```
\r\n
\r\n
Wed, 20 Feb 2008 22:12:21 +0000\r\n

x-amz-copy-source:/amzn-s3-demo-bucket1/flotsam\r\n
/amzn-s3-demo-bucket2/jetsam
```

Amazon S3 将返回以下响应来指定对象的 ETag 及其上次修改的时间。

```
HTTP/1.1 200 OK
x-amz-id-2: Vyaxt7qEbzv34BnSu5hctyyNSlHTYZFMWK4FtzO+iX8JQNyaLdTshL0KxatbaOZt
x-amz-request-id: 6B13C3C5B34AF333
Date: Wed, 20 Feb 2008 22:13:01 +0000

Content-Type: application/xml
Transfer-Encoding: chunked
Connection: close
Server: AmazonS3
<?xml version="1.0" encoding="UTF-8"?>

<CopyObjectResult>
   <LastModified>2008-02-20T22:13:01</LastModified>
   <ETag>"7e9c608af58950deeb370c98608ed097"</ETag>
</CopyObjectResult>
```

使用 AWS CLI

也可以使用 AWS Command Line Interface（AWS CLI）复制 S3 对象。有关更多信息，请参阅 AWS CLI 命令参考 中的 copy-object。

有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的什么是 AWS Command Line Interface？。

移动对象

要移动对象，请使用以下方法。

使用 S3 控制台

> **Note**
>
> - 如果您要移动具有用户定义标签的对象，您还必须拥有 `s3:GetObjectTagging` 权限。如果您要移动的对象没有用户定义的标签，但大小超过 16 MB，您还必须拥有 `s3:GetObjectTagging` 权限。
>
> 如果目标存储桶策略拒绝 `s3:GetObjectTagging` 操作，则将在没有用户定义标签的情况下移动对象，并且您将收到错误。
> - 无法使用 Amazon S3 控制台移动使用客户提供的加密密钥（SSE-C）加密的对象。要移动使用 SSE-C 加密的对象，请使用 AWS CLI、AWS SDK 或 Amazon S3 REST API。
> - 移动文件夹时，请等待移动操作完成，然后再对文件夹进行其它更改。
> - 您不能使用 S3 接入点别名作为 Amazon S3 控制台中移动操作的源或目标。

移动对象

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶，然后选择通用存储桶选项卡。导航到包含待移动对象的 Amazon S3 存储桶或文件夹。
3. 选中要移动的对象名称左侧的复选框。
4. 在操作菜单上，选择移动。
5. 要指定目标路径，请选择 Browse S3（浏览 S3），导航到目标，然后选中目标左侧的复选框。选择右下角的选择目标。

   或者，输入目标路径。
6. 如果未启用存储桶版本控制，则系统可能会要求您确认是否覆盖具有相同名称的现有对象。如果可以覆盖，请选中该复选框并继续。如果要在此存储桶中保留对象的所有版本，请选择 Enable Bucket Versioning（启用存储桶版本控制）。您还可以更新默认加密和对象锁定属性。
7. 选择右下角的移动。Amazon S3 将您的对象移动到目标位置。

> **Note**
>
> - 此操作创建具有更新设置的所有指定对象的副本，更新指定位置的上次修改日期，然后向原始对象添加删除标记。
> - 此操作会更新存储桶版本控制、加密、对象锁定特征和归档对象的元数据。

使用 AWS CLI

您也可以使用 AWS Command Line Interface（AWS CLI）移动 S3 对象。有关更多信息，请参阅 AWS CLI 命令参考 中的 mv。

有关 AWS CLI 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的什么是 AWS Command Line Interface？。

重命名对象

要重命名对象，请使用以下过程。

> **Note**
>
> - 重命名对象将创建具有新的上次修改日期的对象副本，然后在原始对象上添加删除标记。
> - 默认加密的存储桶设置会自动应用于任何未加密的指定对象。
> - 您无法使用 Amazon S3 控制台来重命名采用客户提供的加密密钥（SSE-C）的对象。要重命名使用 SSE-C 加密的对象，请使用 AWS CLI、AWS SDK 或 Amazon S3 REST API 以新名称复制这些对象。
> - 如果此存储桶的 S3 对象所有权使用强制存储桶拥有者设置，则不会复制对象访问控制列表（ACL）。
> - 如果您要重命名具有用户定义标签的对象，您必须拥有 `s3:GetObjectTagging` 权限。如果您要重命名的对象没有用户定义的标签，但大小超过 16 MB，您还必须拥有 `s3:GetObjectTagging` 权限。
>
>   如果目标存储桶策略拒绝 `s3:GetObjectTagging` 操作，则将重命名对象，但将从该对象中移除用户定义的标签，并且您将收到错误。

重命名对象

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶，然后选择通用存储桶选项卡。导航到包含待重命名的对象的 Amazon S3 存储桶或文件夹。
3. 选中要重命名的对象名称左侧的复选框。
4. 在操作菜单中，选择重命名对象。
5. 在新对象名称框中，输入对象的新名称。
6. 选择右下角的保存更改。Amazon S3 随即重命名您的对象。

下载对象

本部分介绍如何从 Amazon S3 存储桶下载对象。使用 Amazon S3，您将这些对象存储在一个或多个存储桶中，每个对象的大小最多为 5 TB。任何未归档的 Amazon S3 对象都可以实时访问。但对于已归档的对象，必须先进行还原，然后才能下载。有关下载归档对象的更多信息，请参阅the section called “下载归档对象”。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API 下载单个对象。要从 S3 下载对象而无需编写任何代码或运行任何命令，请使用 S3 控制台。有关更多信息，请参阅 the section called “下载对象”。

要下载多个对象，请使用 AWS CloudShell、AWS CLI 或 AWS SDK。有关更多信息，请参阅 the section called “下载多个对象”。

如果您需要下载对象的一部分，则需要将 AWS CLI 与额外的参数结合使用，或者使用 REST API 来仅指定要下载的字节。有关更多信息，请参阅 the section called “下载对象的一部分”。

如果您需要下载不属于您的对象，请让对象所有者生成一个允许您下载该对象的预签名 URL。有关更多信息，请参阅 the section called “从另一个 AWS 账户下载对象”。

当您在 AWS 网络之外下载对象时，将收取数据传输费用。在同一个 AWS 区域中，AWS 网络内部的数据传输免费，但任何 GET 请求都将收取费用。有关数据传输成本和数据检索费用的更多信息，请参阅 Amazon S3 定价。

主题

下载对象

您可以使用 Amazon S3 控制台、AWS CLI、AWS SDK 或 REST API 下载单个对象。

使用 S3 控制台

本部分介绍如何使用 Amazon S3 控制台从 S3 存储桶下载对象。

> **Note**
>
> - 一次只能下载一个对象。
> - 如果您使用 Amazon S3 控制台下载对象的键名以句点 (.) 结尾，则该句点将从下载对象的键名中删除。要保留所下载对象名称末尾的句点，您必须使用 AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API。

从 S3 存储桶下载对象

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要从中下载对象的存储桶的名称。
3. 您可以使用以下任一方式从 S3 存储桶下载对象：
   - 选中对象旁边的复选框，然后选择下载。如果您要将对象下载到特定文件夹，请在操作菜单中，选择下载为。
   - 如果要下载对象的特定版本，请打开显示版本（位于搜索框旁边）。选中所需对象版本旁边的复选框，然后选择下载。如果您要将对象下载到特定文件夹，请在操作菜单中，选择下载为。

使用 AWS CLI

以下 `get-object` 示例将向您展示如何使用 AWS CLI 从 Amazon S3 下载对象。此命令从存储桶 *amzn-s3-demo-bucket1* 获取对象 *folder/my_image*。该对象将下载到名为 *my_downloaded_image* 的文件中。

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key folder/
my_image my_downloaded_image
```

有关更多信息和示例，请参阅 AWS CLI 命令参考中的 get-object。

使用 AWS SDK

有关如何使用 AWS SDK 下载对象的示例，请参阅将 GetObject 与 AWS SDK 或 CLI 配合使用。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

使用 REST API

您可以使用 REST API 从 Amazon S3 中检索对象。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 GetObject。

下载多个对象

您可以使用 AWS CloudShell、AWS CLI 或 AWS SDK 下载多个对象。

在 AWS Management Console 中使用 AWS CloudShell

AWS CloudShell 是一个已经事先完成身份验证的浏览器式 Shell，您可以直接从AWS Management Console启动它。

有关 AWS CloudShell 的更多信息，请参阅《AWS CloudShell User Guide》中的What is CloudShell?。

> **Important**
>
> 使用 AWS CloudShell，您的主目录的存储空间最高为每个 AWS 区域 1 GB。因此，您无法将存储桶与总大小超过此数量的对象同步。有关更多限制，请参阅《AWS CloudShell User Guide》中的 eService quotas and restrictions。

使用 AWS CloudShell 下载对象

1. 登录 AWS Management Console，然后通过以下网址打开 CloudShell 控制台：https://console.aws.amazon.com/cloudshell/.
2. 运行以下命令，将存储桶中的对象同步到 CloudShell。以下命令同步名为 *amzn-s3-demo-bucket1* 的存储桶中的对象，并在 CloudShell 中创建一个名为 *temp* 的文件夹。CloudShell 会将您的对象同步到这个文件夹。要使用此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3 sync s3://amzn-s3-demo-bucket1 ./temp
```

> **Note**
>
> 要执行模式匹配以排除或包含特定对象，您可以在 sync 命令中使用 --exclude "*value*" 和 --include "*value*" 参数。

3. 运行以下命令，将名为 *temp* 的文件夹中的对象压缩到名为 *temp.zip* 的文件中。

```
zip temp.zip -r temp/
```

4. 选择操作，然后选择下载文件。
5. 输入文件名 **temp.zip**，然后选择下载。
6. （可选）删除同步到 CloudShell 中 *temp* 文件夹的 *temp.zip* 文件和对象。使用 AWS CloudShell，您可在每个 AWS 区域中拥有最高 1 GB 的持久性存储。

   您可以使用此示例命令来删除您的 .zip 文件和文件夹。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
rm temp.zip && rm -rf temp/
```

使用 AWS CLI

此示例说明如何使用 AWS CLI 下载指定目录或前缀下的所有文件或对象。此命令将存储桶 *amzn-s3-demo-bucket1* 中的所有对象复制到您的当前目录。要使用此示例命令，请使用您的存储桶名称代替 *amzn-s3-demo-bucket1*。

```
aws s3 cp s3://amzn-s3-demo-bucket1 . --recursive
```

以下命令将存储桶 *amzn-s3-demo-bucket1* 中前缀 *logs* 下的所有对象下载到您的当前目录。它还使用 --exclude 和 --include 参数仅复制带有后缀 *.log* 的对象。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3 cp s3://amzn-s3-demo-bucket1/logs/ . --recursive --exclude "*" --include "*.log"
```

有关更多信息和示例，请参阅 AWS CLI 命令参考中的 cp。

使用 AWS SDK

有关如何使用 AWS SDK 下载 Amazon S3 存储桶中对象的示例，请参阅将 Amazon Simple Storage Service（Amazon S3）桶中的所有对象下载到本地目录。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

下载对象的一部分

您可以使用 AWS CLI 或 REST API 下载对象的一部分。为此，您需要使用额外的参数来指定要下载对象的哪一部分。

使用 AWS CLI

以下示例命令在名为 *amzn-s3-demo-bucket1* 的存储桶中，对名为 *folder/my_data* 的对象中的字节范围执行 GET 请求。在请求中，字节范围必须以 bytes= 为前缀。部分对象将下载到名为 *my_data_range* 的输出文件中。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key folder/my_data --range
 bytes=0-500 my_data_range
```

有关更多信息和示例，请参阅 AWS CLI 命令参考中的 get-object。

有关 HTTP Range 标头的更多信息，请参阅 RFC 编辑器网站上的 RFC 9110。

> **Note**
>
> Amazon S3 不支持在单个 GET 请求中检索多个数据范围。

使用 REST API

您可以在 REST API 中使用 `partNumber` 和 `Range` 参数，从 Amazon S3 检索对象部分。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 GetObject。

从另一个 AWS 账户下载对象

您可以使用预签名 URL 授予对对象的限时访问权限，而不更新存储桶策略。

预签名 URL 可以在浏览器中输入，或者由程序用来下载对象。URL 使用的凭证是生成该 URL 的 AWS 用户的凭证。创建 URL 后，在该 URL 过期之前，任何拥有预签名 URL 的人都可以下载相应的对象。

在 S3 控制台中使用预签名 URL

您可以使用 Amazon S3 控制台，按照以下步骤生成预签名 URL 来共享对象。使用控制台时，预签名 URL 的最长过期时间为自创建时起 12 小时。

使用 Amazon S3 控制台生成预签名 URL

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（存储桶）列表中，请选择包含要为其生成预签名 URL 的对象的存储桶的名称。
4. 在 Objects（对象）列表中，选择要为其生成预签名 URL 的对象。
5. 在对象操作菜单上，请选择使用预签名 URL 共享。
6. 指定您希望的预签名 URL 有效时间长度。
7. 请选择 Create presigned URL（创建预签名 URL）。
8. 出现确认消息时，URL 将自动复制到剪贴板。如果您需要再次复制预签名 URL，您将看到一个按钮，用于复制该 URL。
9. 要下载对象，请将 URL 粘贴到任何浏览器中，此时机会尝试下载该对象。

有关预签名 URL 和其他创建预签名 URL 方法的更多信息，请参阅使用预签名 URL。

下载归档对象

要降低不经常访问的对象的存储成本，您可以对此类对象进行归档。当您归档对象时，它会被移动到低成本存储中，这意味着您无法实时访问它。要下载归档对象，您必须先还原该对象。

根据存储类的不同，您可以在几分钟或几小时内还原归档对象。您可以使用 Amazon S3 控制台、S3 批量操作、Amazon S3 REST API、AWS SDK 和 AWS Command Line Interface（AWS CLI）还原归档的对象。

有关说明，请参阅 恢复已归档的对象。您可在还原归档对象后下载它。

下载对象故障排查

当您尝试从 Amazon S3 下载对象时，权限不足、错误的存储桶或错误的 AWS Identity and Access Management（IAM）用户策略会导致出错。这些问题通常会导致访问被拒绝（403 禁止）错误，此时 Amazon S3 会禁止访问资源。

有关拒绝访问（403 禁止）错误的常见原因，请参阅排查 Amazon S3 中的拒绝访问（403 Forbidden）错误。

检查对象完整性

Amazon S3 使用校验和值验证您上传到 Amazon S3 或从 Amazon S3 下载的数据的完整性。此外，您可以请求为存储在 Amazon S3 中的任何对象计算另一个校验和值。您可以从几种校验和算法中进行选择，以便在上传或复制数据时使用。Amazon S3 使用此算法计算额外的校验和值并将其存储为对象元数据的一部分。要详细了解如何使用其他校验和来验证数据完整性，请参阅教程：使用其他校验和检查 Amazon S3 中数据的完整性。

在上传对象时，您可以选择将预先计算的校验和作为请求的一部分包含在内。Amazon S3 将提供的校验和与它使用指定算法计算出的校验和进行比较。如果这两个值不匹配，Amazon S3 会报告错误。

使用支持的校验和算法

Amazon S3 为您提供了选择校验和算法的选项，该算法用于在上传或下载期间验证数据。您可以选择以下安全哈希算法 (SHA) 或循环冗余校验 (CRC) 数据完整性检查算法之一：

- CRC32
- CRC32C
- SHA-1
- SHA-256

在上传对象时，可指定要使用的算法：

- 使用 AWS Management Console 时，选择要使用的校验和算法。当您这样做时，可以选择指定对象的校验和值。当 Amazon S3 收到对象时，它使用您指定的算法计算校验和。如果这两个校验和值不匹配，Amazon S3 会生成错误。
- 在使用 SDK 时，您可以将 `ChecksumAlgorithm` 参数的值设置为您希望 Amazon S3 在计算校验和时使用的算法。Amazon S3 会自动计算校验和值。
- 在使用 REST API 时，不使用 `x-amz-sdk-checksum-algorithm` 参数，而是使用一个特定于算法的标头（例如，`x-amz-checksum-crc32`）。

有关上传对象的更多信息，请参阅上传对象。

要将这些校验和值中的任何一个应用于已上传到 Amazon S3 的对象，您可以复制该对象。复制对象时，可以指定是使用现有校验和算法还是使用新校验和算法。在使用任何受支持的机制复制对象（包括 S3 批量操作）时，您可以指定校验和算法。有关 S3 批量操作的更多信息，请参阅 对 Amazon S3 对象执行大规模批量操作。

> **Important**
>
> 如果您结合其他校验和使用分段上传，则分段编号必须是连续的编号。使用其他校验和时，如果您尝试使用非连续分段编号完成分段上传请求，Amazon S3 将生成 HTTP `500 Internal Server Error` 错误。

上传对象后，您可以获取校验和值，并将其与预先计算的或之前存储的使用相同算法计算出的校验和值进行比较。

使用 S3 控制台

要了解有关使用控制台和指定在上传对象时要使用的校验和算法的更多信息，请参阅上传对象和教程：使用其他校验和检查 Amazon S3 中数据的完整性。

使用 AWS SDK

以下示例显示了如何使用 AWS SDK 通过分段上传来上传大文件、下载大文件和验证分段上传文件，所有这些都使用 SHA-256 进行文件验证。

Java

Example 示例：使用 SHA-256 上传、下载和验证大文件

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.PutObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.Tag;
import software.amazon.awssdk.services.s3.model.Tagging;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.ByteBuffer;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;

public class LargeObjectValidation {
    private static String FILE_NAME = "sample.file";
    private static String BUCKET = "sample-bucket";
```

```
        //Optional, if you want a method of storing the full multipart object
 checksum in S3.
        private static String CHECKSUM_TAG_KEYNAME = "fullObjectChecksum";
        //If you have existing full-object checksums that you need to validate
 against, you can do the full object validation on a sequential upload.
        private static String SHA256_FILE_BYTES = "htCM5g7ZNdoSw8bN/
mkgiAhXt5MFoVowVg+LE9aIQmI=";
        //Example Chunk Size - this must be greater than or equal to 5MB.
        private static int CHUNK_SIZE = 5 * 1024 * 1024;

        public static void main(String[] args) {
            S3Client s3Client = S3Client.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(new AwsCredentialsProvider() {
                        @Override
                        public AwsCredentials resolveCredentials() {
                            return new AwsCredentials() {
                                @Override
                                public String accessKeyId() {
                                    return Constants.ACCESS_KEY;
                                }

                                @Override
                                public String secretAccessKey() {
                                    return Constants.SECRET;
                                }
                            };
                        }
                    })
                    .build();
            uploadLargeFileBracketedByChecksum(s3Client);
            downloadLargeFileBracketedByChecksum(s3Client);
            validateExistingFileAgainstS3Checksum(s3Client);
        }

        public static void uploadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting uploading file validation");
            File file = new File(FILE_NAME);
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256 = MessageDigest.getInstance("SHA-256");
                CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                        .bucket(BUCKET)
                        .key(FILE_NAME)
```

```
                        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
                        .build();
                CreateMultipartUploadResponse createdUpload =
s3Client.createMultipartUpload(createMultipartUploadRequest);
                List<CompletedPart> completedParts = new ArrayList<CompletedPart>();
                int partNumber = 1;
                byte[] buffer = new byte[CHUNK_SIZE];
                int read = in.read(buffer);
                while (read != -1) {
                    UploadPartRequest uploadPartRequest =
UploadPartRequest.builder()

.partNumber(partNumber).uploadId(createdUpload.uploadId()).key(FILE_NAME).bucket(BUCKET).ch
                    UploadPartResponse uploadedPart =
s3Client.uploadPart(uploadPartRequest,
RequestBody.fromByteBuffer(ByteBuffer.wrap(buffer, 0, read)));
                    CompletedPart part =
CompletedPart.builder().partNumber(partNumber).checksumSHA256(uploadedPart.checksumSHA256()
                    completedParts.add(part);
                    sha256.update(buffer, 0, read);
                    read = in.read(buffer);
                    partNumber++;
                }
                String fullObjectChecksum =
Base64.getEncoder().encodeToString(sha256.digest());
                if (!fullObjectChecksum.equals(SHA256_FILE_BYTES)) {
                    //Because the SHA256 is uploaded after the part is uploaded; the
upload is bracketed and the full object can be fully validated.

s3Client.abortMultipartUpload(AbortMultipartUploadRequest.builder().bucket(BUCKET).key(FILE
                    throw new IOException("Byte mismatch between stored checksum and
upload, do not proceed with upload and cleanup");
                }
                CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder().parts(completedParts).build();
                CompleteMultipartUploadResponse completedUploadResponse =
s3Client.completeMultipartUpload(

CompleteMultipartUploadRequest.builder().bucket(BUCKET).key(FILE_NAME).uploadId(createdUplo
                Tag checksumTag =
Tag.builder().key(CHECKSUM_TAG_KEYNAME).value(fullObjectChecksum).build();
                //Optionally, if you need the full object checksum stored with the
file; you could add it as a tag after completion.
```

```
s3Client.putObjectTagging(PutObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).t
            } catch (IOException | NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
            GetObjectAttributesResponse
                    objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
            System.out.println(objectAttributes.objectParts().parts());
            System.out.println(objectAttributes.checksum().checksumSHA256());
        }

        public static void downloadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting downloading file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            try (OutputStream out = new FileOutputStream(file)) {
                GetObjectAttributesResponse
                        objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                        .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
                //Optionally if you need the full object checksum, you can grab a
tag you added on the upload
                List<Tag> objectTags =
s3Client.getObjectTagging(GetObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).b
                String fullObjectChecksum = null;
                for (Tag objectTag : objectTags) {
                    if (objectTag.key().equals(CHECKSUM_TAG_KEYNAME)) {
                        fullObjectChecksum = objectTag.value();
                        break;
                    }
                }
                MessageDigest sha256FullObject =
MessageDigest.getInstance("SHA-256");
                MessageDigest sha256ChecksumOfChecksums =
MessageDigest.getInstance("SHA-256");

                //If you retrieve the object in parts, and set the ChecksumMode to
enabled, the SDK will automatically validate the part checksum
                for (int partNumber = 1; partNumber <=
objectAttributes.objectParts().totalPartsCount(); partNumber++) {
                    MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
```

```
                    ResponseInputStream<GetObjectResponse> response =
 s3Client.getObject(GetObjectRequest.builder().bucket(BUCKET).key(FILE_NAME).partNumber(part
                    GetObjectResponse getObjectResponse = response.response();
                    byte[] buffer = new byte[CHUNK_SIZE];
                    int read = response.read(buffer);
                    while (read != -1) {
                        out.write(buffer, 0, read);
                        sha256FullObject.update(buffer, 0, read);
                        sha256Part.update(buffer, 0, read);
                        read = response.read(buffer);
                    }
                    byte[] sha256PartBytes = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(sha256PartBytes);
                    //Optionally, you can do an additional manual validation again
 the part checksum if needed in addition to the SDK check
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(sha256PartBytes);
                    String base64PartChecksumFromObjectAttributes =
 objectAttributes.objectParts().parts().get(partNumber - 1).checksumSHA256();
                    if (!
base64PartChecksum.equals(getObjectResponse.checksumSHA256()) || !
base64PartChecksum.equals(base64PartChecksumFromObjectAttributes)) {
                        throw new IOException("Part checksum didn't match for the
 part");
                    }
                    System.out.println(partNumber + " " + base64PartChecksum);
                }
                //Before finalizing, do the final checksum validation.
                String base64FullObject =
 Base64.getEncoder().encodeToString(sha256FullObject.digest());
                String base64ChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                if (fullObjectChecksum != null && !
fullObjectChecksum.equals(base64FullObject)) {
                    throw new IOException("Failed checksum validation for full
 object");
                }
                System.out.println(fullObjectChecksum);
                String base64ChecksumOfChecksumFromAttributes =
 objectAttributes.checksum().checksumSHA256();
                if (base64ChecksumOfChecksumFromAttributes != null && !
base64ChecksumOfChecksums.equals(base64ChecksumOfChecksumFromAttributes)) {
                    throw new IOException("Failed checksum validation for full
 object checksum of checksums");
```

```
                }
                System.out.println(base64ChecksumOfChecksumFromAttributes);
                out.flush();
            } catch (IOException | NoSuchAlgorithmException e) {
                //Cleanup bad file
                file.delete();
                e.printStackTrace();
            }
        }

        public static void validateExistingFileAgainstS3Checksum(S3Client s3Client)
{
            System.out.println("Starting existing file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            GetObjectAttributesResponse
                    objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256ChecksumOfChecksums =
MessageDigest.getInstance("SHA-256");
                MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                byte[] buffer = new byte[CHUNK_SIZE];
                int currentPart = 0;
                int partBreak =
objectAttributes.objectParts().parts().get(currentPart).size();
                int totalRead = 0;
                int read = in.read(buffer);
                while (read != -1) {
                    totalRead += read;
                    if (totalRead >= partBreak) {
                        int difference = totalRead - partBreak;
                        byte[] partChecksum;
                        if (totalRead != partBreak) {
                            sha256Part.update(buffer, 0, read - difference);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                            sha256Part.update(buffer, read - difference,
difference);
                        } else {
                            sha256Part.update(buffer, 0, read);
                            partChecksum = sha256Part.digest();
```

```
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                        }
                        String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                        if (!
base64PartChecksum.equals(objectAttributes.objectParts().parts().get(currentPart).checksumSH
 {
                            throw new IOException("Part checksum didn't match S3");
                        }
                        currentPart++;
                        System.out.println(currentPart + " " + base64PartChecksum);
                        if (currentPart <
 objectAttributes.objectParts().totalPartsCount()) {
                            partBreak +=
 objectAttributes.objectParts().parts().get(currentPart - 1).size();
                        }
                    } else {
                        sha256Part.update(buffer, 0, read);
                    }
                    read = in.read(buffer);
                }
                if (currentPart != objectAttributes.objectParts().totalPartsCount())
 {
                    currentPart++;
                    byte[] partChecksum = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(partChecksum);
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                    System.out.println(currentPart + " " + base64PartChecksum);
                }

                String base64CalculatedChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                System.out.println(base64CalculatedChecksumOfChecksums);
                System.out.println(objectAttributes.checksum().checksumSHA256());
                if (!
base64CalculatedChecksumOfChecksums.equals(objectAttributes.checksum().checksumSHA256()))
 {
                    throw new IOException("Full object checksum of checksums don't
 match S3");
                }

            } catch (IOException | NoSuchAlgorithmException e) {
```

```
                e.printStackTrace();
            }
        }
    }
```

使用 REST API

您可以发送 REST 请求以上传具有校验和值的对象，通过 PutObject 验证数据的完整性。您还可以使用 GetObject 或 HeadObject 检索对象的校验和值。

使用 AWS CLI

您可以发送 PUT 请求，在单个操作中上传最大 5 GB 的对象。有关更多信息，请参阅《AWS CLI 命令参考》中的 `PutObject`。您还可以使用 `get-object` 和 `head-object` 检索已上传对象的校验和以验证数据的完整性。

有关信息，请参阅《AWS Command Line Interface 用户指南》中的 Amazon S3 CLI FAQ。

在上传对象时使用 Content-M5

上传后验证对象完整性的另一种方法是在上传对象时提供其 MD5 摘要。如果您计算对象的 MD5 摘要，则可以使用 `Content-MD5` 标头向 PUT 命令提供摘要。

上传对象后，Amazon S3 会计算对象的 MD5 摘要，并将其与您提供的值进行比较。只有在两个摘要匹配的情况下，请求才会成功。

不要求提供 MD5 摘要，但您可以在上传过程中使用它来验证对象的完整性。

使用 Content-MD5 和 ETag 验证上传的对象

对象的实体标签 (ETag) 表示该对象的特定版本。请注意，ETag 仅反映对对象内容的更改，而不反映对对象元数据的更改。如果只有对象的元数据发生变化，eTag 将保持不变。

根据对象的不同，对象的 ETag 可能是对象数据的 MD5 摘要：

- 如果对象是由 `PutObject`、`PostObject` 或 `CopyObject` 操作创建或通过 AWS Management Console创建，并且该对象还是纯文本或使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）进行加密的，则该对象的 ETag 将是其对象数据的 MD5 摘要。
- 如果对象是由 `PutObject`、`PostObject` 或 `CopyObject` 操作创建或通过 AWS Management Console创建，并且该对象使用客户提供的密钥 (SSE-C) 或 AWS Key Management Service (AWS

KMS) 密钥 (SSE-KMS) 通过服务器端加密进行加密，则该对象的 ETag 将不是其对象数据的 MD5 摘要。

- 如果对象由 `Multipart Upload` 或 `Part Copy` 操作创建，无论使用哪种加密方法，对象的 ETag 都不是 MD5 摘要。如果对象大于 16 MB，则 AWS Management Console会作为分段上传来上传或复制该对象，因此 ETag 不是 MD5 摘要。

对于 ETag 是对象的 `Content-MD5` 摘要的对象，您可以将对象的 ETag 值与计算出的或之前存储的 `Content-MD5` 摘要进行比较。

使用尾随校验和

将对象上传到 Amazon S3 时，您可以为对象提供预先计算的校验和，也可以使用 AWS SDK 代表您自动创建尾随校验和。如果您决定使用尾随校验和，Amazon S3 会使用您指定的算法自动生成校验和，并在上传期间使用它验证对象的完整性。

要使用 AWS SDK 时创建尾随校验和，请使用您首选的算法填充 `ChecksumAlgorithm` 参数。SDK 使用该算法计算对象（或对象分段）的校验和，并自动将其附加到上传请求的末尾。这种行为可以节省您的时间，因为 Amazon S3 一次性执行数据的验证和上传。

> **Important**
>
> 如果您使用的是 S3 对象 Lambda，则对 S3 对象 Lambda 的所有请求都使用 `s3-object-lambda` 而不是 `s3`。此行为会影响尾随校验和值的签名。有关 S3 对象 Lambda 的更多信息，请参阅 使用 S3 对象 Lambda 转换对象。

使用分段级别墅校验和进行分段上传

将对象上传到 Amazon S3 时，它们可以作为单个对象上传，也可以通过分段上传过程进行上传。大于 16 MB 且通过控制台上传的对象将使用分段上传自动上传。有关分段上传的更多信息，请参阅 使用分段上传来上传和复制对象。

当对象作为分段上传进行上传时，该对象的 ETag 不是整个对象的 MD5 摘要。Amazon S3 会在上传时计算每个分段的 MD5 摘要。MD5 摘要用于确定最终对象的 ETag。Amazon S3 将 MD5 摘要的字节串联在一起，然后计算这些串联值的 MD5 摘要。创建 ETag 的最后一步是 Amazon S3 在末尾添加一个短横线和分段总数。

例如，考虑使用 ETag 为 C9A5A6878D97B48CC965C1E41859F034-14 的分段上传的对象上传。在本例中，C9A5A6878D97B48CC965C1E41859F034 是串联在一起的所有摘要的 MD5 摘要。-14 表示有 14 个分段与此对象的分段上传相关联。

如果您已为多分段对象启用了其他校验和值，Amazon S3 将使用指定的校验和算法计算每个分段的校验和。已完成对象的校验和的计算方法与 Amazon S3 计算分段上传的 MD5 摘要的方法相同。您可以使用此校验和验证对象的完整性。

要检索有关对象的信息，包括组成整个对象的分段数，可以使用 GetObjectAttributes 操作。借助其他校验和，您还可以恢复每个分段的信息，包括每个分段的校验和值。

对于已完成的分段上传，您可以使用 GetObject 或 HeadObject 操作，并指定与单个分段相符的分段编号或字节范围，来获取单个分段的校验和。如果您想检索仍在进行的分段上传的各个分段的校验和值，可以使用 ListParts。

由于 Amazon S3 计算分段对象的校验和的方式，复制对象时，对象的校验和值可能会发生变化。如果您使用的是SDK 或 REST API，并且调用 CopyObject，Amazon S3 可以复制大小不超过 CopyObject API 操作限制的任何对象。无论对象是在单个请求中上传还是作为分段上传的一部分上传，Amazon S3 都可以作为单个操作执行此复制。使用 copy 命令，对象的校验和是完整对象的直接校验和。如果对象最初是使用分段上传进行上传的，那么即使数据没有变化，校验和值也会发生变化。

> **Note**
>
> 大于 CopyObject API 操作的大小限制的对象必须使用分段复制命令。

> **Important**
>
> 当您使用 AWS Management Console执行某些操作时，如果对象大于 16 MB，则 Amazon S3 将使用分段上传。在这种情况下，校验和不是完整对象的直接校验和，而是基于每个分段的校验和值的计算值。
> 例如，考虑一个大小为 100 MB 的对象，您使用 REST API 作为单个分段直接上传该对象。在这种情况下，校验和是整个对象的校验和。如果您稍后使用控制台重命名该对象、复制对象、更改存储类或编辑元数据，Amazon S3 将使用分段上传功能来更新对象。因此，Amazon S3 为对象创建一个新的校验和值，该值是根据各个分段的校验和值计算的。
> 前面的控制台操作列表并不是您可以在 AWS Management Console 中执行的所有可能操作的完整列表，这些操作会导致 Amazon S3 使用分段上传功能更新对象。请记住，无论何时使用控制台对大小超过 16 MB 的对象执行操作，校验和值都可能不是整个对象的校验和。

删除 Amazon S3 对象

您可以使用 Amazon S3 控制台、AWS SDK、AWS Command Line Interface (AWS CLI) 或 REST API 直接从 Amazon S3 删除一个或多个对象。由于 S3 存储桶中的所有对象都会产生存储费用，因此您应从中删除不再需要的对象。例如，如果您正在收集日志文件，最好在不再需要这些文件时将其删除。您可以将生命周期规则设置为自动删除对象 (如日志文件)。有关更多信息，请参阅 the section called “设置生命周期配置”。

有关 Amazon S3 特征和定价的信息，请参阅 Amazon S3 定价。

删除对象时，您可以使用以下 API 选项：

- 删除单个对象 – Amazon S3 提供了 `DELETE`（`DeleteObject`）API 操作，使您能够删除单个 HTTP 请求中的一个对象。
- 删除多个对象 – Amazon S3 提供了多对象删除（`DeleteObjects`）API 操作，使您能够在单个 HTTP 请求中删除多达 1000 个对象。

从未启用版本控制的存储桶中删除对象时，只需提供对象键名称。但是，从启用版本控制的存储桶中删除对象时，您可以选择提供对象的版本 ID，来删除该对象的特定版本。

以编程方式从启用版本控制的存储桶中删除对象

如果存储桶已启用版本控制，则存储桶中可能存在同一对象的多个版本。使用启用了版本控制的存储桶时，删除 API 操作将启用以下选项：

- 指定不受版本控制的删除请求 – 仅指定对象的键，而不指定版本 ID。在此情况下，Amazon S3 将创建一个删除标记并返回它在响应中的版本 ID。这将使您的对象从存储桶中消失。有关对象版本控制和删除标记概念的信息，请参阅 在 S3 存储桶中使用版本控制。
- 指定受版本控制的删除请求 – 不仅可以指定键，还可以指定版本 ID。在此情况下，可能会出现以下两种结果：
  - 如果版本 ID 映射到特定的对象版本，则 Amazon S3 将删除该特定版本的对象。
  - 如果版本 ID 映射到对象的删除标记，则 Amazon S3 将删除该删除标记。这将使您的对象重新出现在存储桶中。

从启用了 MFA 的存储桶中删除对象

从启用了多重身份验证（MFA）的存储桶中删除对象时，请注意以下内容：

- 如果您提供了无效的 MFA 令牌，请求将始终失败。
- 如果您拥有一个启用了 MFA 的存储桶，并且发送了一个受版本控制的删除请求（您提供了对象键和版本 ID），若您不能提供有效的 MFA 令牌，请求将失败。此外，对启用了 MFA 的存储桶使用多对象删除 API 操作时，如果任意删除是受版本控制的删除请求（即您指定了对象键和版本 ID），若您不能提供有效的 MFA 令牌，则整个请求将失败。

但是，在下面的情况下，请求将成功：

- 如果您拥有一个启用了 MFA 的存储桶，并发出一个不受版本控制的删除请求（不删除受版本控制的对象），且您未提供 MFA 令牌，删除会成功。
- 如果您发出一个多对象删除请求，此请求指定从一个启用了 MFA 的存储桶中仅删除不受版本控制的对象，且您未提供 MFA 令牌，这些删除会成功。

有关 MFA 删除的信息，请参阅配置 MFA 删除。

主题

删除单个对象

您可以使用 Amazon S3 控制台或 DELETE API 从 S3 存储桶中删除单个现有对象。有关删除 Amazon S3 中的对象的更多信息，请参阅删除 Amazon S3 对象。

由于 S3 存储桶中的所有对象都会产生存储费用，因此您应从中删除不再需要的对象。例如，如果您正在收集日志文件，最好在不再需要这些文件时将其删除。您可以将生命周期规则设置为自动删除对象(如日志文件)。有关更多信息，请参阅 the section called “设置生命周期配置”。

有关 Amazon S3 特征和定价的信息，请参阅 Amazon S3 定价。

使用 S3 控制台

请按照以下步骤使用 Amazon S3 控制台从存储桶中删除单个对象。

> **Warning**
>
> 当您在 Amazon S3 控制台中永久删除对象或指定的对象版本时，删除操作无法撤销。

删除已启用或暂停版本控制的对象

> **Note**
>
> 如果已暂停版本控制的桶中某个对象的版本 ID 标记为 NULL，则 S3 会永久删除该对象，因为不存在之前的版本。然而，如果在暂停了版本控制的桶中为对象列出了有效的版本 ID，则 S3 为已删除的对象创建删除标记，同时保留该对象的先前版本。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket name（存储桶名称）列表中，请选择要从中删除对象的存储桶的名称。
3. 选择对象，然后选择删除。
4. 要确认删除指定的对象下的对象列表，请在删除对象？文本框中输入 **`delete`**。

永久删除已启用版本控制的桶中的特定对象版本

> **Warning**
>
> 当您永久删除 Amazon S3 中的特定对象版本时，删除操作无法撤销。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket name（存储桶名称）列表中，请选择要从中删除对象的存储桶的名称。
3. 选择您要删除的对象。
4. 选择显示版本开关。
5. 选择对象版本，然后选择删除。
6. 要确认永久删除指定的对象下列出的特定对象版本，请在删除对象？文本框中输入 Permanently delete。Amazon S3 永久删除特定的对象版本。

永久删除未启用版本控制的 Amazon S3 存储桶中的对象

> **Warning**
>
> 当您永久删除 Amazon S3 中的对象时，删除操作无法撤销。此外，对于任何未启用版本控制的桶，删除操作都是永久性的。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket name（存储桶名称）列表中，请选择要从中删除对象的存储桶的名称。
3. 选择对象，然后选择删除。
4. 要确认永久删除指定的对象下列出的对象，请在删除对象？文本框中输入 permanently delete。

> **Note**
>
> 如果您在删除对象时遇到任何问题，请参阅我想永久删除受版本控制的对象。

使用 AWS 软件开发工具包

以下示例展示了如何使用 AWS SDK 从存储桶中删除对象。有关更多信息，请参阅 Amazon Simple Storage Service API 参考中的 DELETE Object

如果已对存储桶启用 S3 版本控制，您将可使用以下选项：

- 通过指定版本 ID 来删除特定对象版本。
- 删除对象而不指定版本 ID，在这种情况下，Amazon S3 将向对象添加一个删除标记。

有关 S3 版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。

Java

Example 示例 1：删除对象（不受版本控制的存储桶）

以下示例假定存储桶不受版本控制且对象没有任何版本 ID。在删除请求中，您仅指定对象键而不是版本 ID。

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

import java.io.IOException;

public class DeleteObjectNonVersionedBucket {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(bucketName, keyName));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 示例 2：删除对象（受版本控制的存储桶）

以下示例将删除受版本控制的存储桶中的对象。该示例通过指定对象键名和版本 ID 来删除特定对象版本。

本示例执行以下操作：

1. 向存储桶添加示例对象。Amazon S3 会返回新添加对象的版本 ID。该示例将在删除请求中使用此版本 ID。
2. 通过指定对象键名和版本 ID 来删除对象版本。如果对象没有其他版本，则 Amazon S3 将完全删除对象。否则，Amazon S3 仅删除指定版本。

> **Note**
>
> 可以通过发送 `ListVersions` 请求来获取对象的版本 ID。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.DeleteVersionRequest;
import com.amazonaws.services.s3.model.PutObjectResult;

import java.io.IOException;

public class DeleteObjectVersionEnabledBucket {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ****";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Check to ensure that the bucket is versioning-enabled.
            String bucketVersionStatus =
 s3Client.getBucketVersioningConfiguration(bucketName).getStatus();
```

```
            if (!bucketVersionStatus.equals(BucketVersioningConfiguration.ENABLED))
{
                System.out.printf("Bucket %s is not versioning-enabled.",
bucketName);
            } else {
                // Add an object.
                PutObjectResult putResult = s3Client.putObject(bucketName, keyName,
                        "Sample content for deletion example.");
                System.out.printf("Object %s added to bucket %s\n", keyName,
bucketName);

                // Delete the version of the object that we just created.
                System.out.println("Deleting versioned object " + keyName);
                s3Client.deleteVersion(new DeleteVersionRequest(bucketName, keyName,
putResult.getVersionId()));
                System.out.printf("Object %s, version %s deleted\n", keyName,
putResult.getVersionId());
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

以下示例演示如何删除受版本控制和不受版本控制的存储桶中的对象。有关 S3 版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。

Example 删除不受版本控制的存储桶中的对象

以下 C# 示例将删除不受版本控制的存储桶中的对象。该示例假定对象没有版本 ID，因此您未指定版本 ID。您仅指定了对象键。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteObjectNonVersionedBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            DeleteObjectNonVersionedBucketAsync().Wait();
        }
        private static async Task DeleteObjectNonVersionedBucketAsync()
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(deleteObjectRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
            catch (Exception e)
            {
```

```
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
        }
    }
}
```

Example 删除受版本控制的存储桶中的对象

以下 C# 示例将删除受版本控制的存储桶中的对象。它将通过指定对象键名和版本 ID 来删除对象的特定版本。

代码将执行以下任务：

1. 对指定的存储桶启用版本控制（如果已启用 S3 版本控制，则此操作无效）。
2. 向存储桶添加示例对象。作为响应，Amazon S3 将返回新添加的对象的版本 ID。该示例将在删除请求中使用此版本 ID。
3. 通过指定对象键名和版本 ID 来删除示例对象。

> **Note**
>
> 还可以通过发送 `ListVersions` 请求来获取对象的版本 ID。
>
> ```
> var listResponse = client.ListVersions(new ListVersionsRequest { BucketName
>  = bucketName, Prefix = keyName });
> ```

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteObjectVersion
    {
```

```
        private const string bucketName = "*** versioning-enabled bucket name ***";
        private const string keyName = "*** Object Key Name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateAndDeleteObjectVersionAsync().Wait();
        }

        private static async Task CreateAndDeleteObjectVersionAsync()
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(keyName);

                // Delete the object by specifying an object key and a version ID.
                DeleteObjectRequest request = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID
                };
                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
deleting an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
deleting an object", e.Message);
            }
        }

        static async Task<string> PutAnObject(string objectKey)
        {
```

```
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!"
            };
            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
}
```

PHP

本示例演示如何使用第 3 版 AWS SDK for PHP 中的类删除不受版本控制的存储桶中的对象。有关从受版本控制的存储桶中删除对象的信息，请参阅 使用 REST API。

有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

下面的 PHP 示例将删除存储桶中的对象。由于此示例演示如何删除不受版本控制的存储桶中的对象，因此它在删除请求中仅提供存储桶名称和对象键（而不是版本 ID）。

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// 1. Delete the object from the bucket.
try
{
    echo 'Attempting to delete ' . $keyname . '...' . PHP_EOL;
```

```
    $result = $s3->deleteObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);

    if ($result['DeleteMarker'])
    {
        echo $keyname . ' was deleted or does not exist.' . PHP_EOL;
    } else {
        exit('Error: ' . $keyname . ' was not deleted.' . PHP_EOL);
    }
}
catch (S3Exception $e) {
    exit('Error: ' . $e->getAwsErrorMessage() . PHP_EOL);
}

// 2. Check to see if the object was deleted.
try
{
    echo 'Checking to see if ' . $keyname . ' still exists...' . PHP_EOL;

    $result = $s3->getObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);

    echo 'Error: ' . $keyname . ' still exists.';
}
catch (S3Exception $e) {
    exit($e->getAwsErrorMessage());
}
```

Javascript

```
import { DeleteObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js" // Helper function that creates Amazon
 S3 service client module.

export const bucketParams = { Bucket: "BUCKET_NAME", Key: "KEY" };

export const run = async () => {
  try {
    const data = await s3Client.send(new DeleteObjectCommand(bucketParams));
```

```
    console.log("Success. Object deleted.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

使用 AWS CLI

要为每个请求删除一个对象，请使用 DELETE API。有关更多信息，请参阅 DELETE Object。有关使用 CLI 删除对象的更多信息，请参阅 delete-object。

使用 REST API

您可以使用 AWS SDK 删除对象。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。有关更多信息，请参阅 Amazon Simple Storage Service API 参考中的 DELETE Object。

删除多个对象

由于 S3 存储桶中的所有对象都会产生存储费用，因此您应从中删除不再需要的对象。例如，如果您正在收集日志文件，最好在不再需要这些文件时将其删除。您可以将生命周期规则设置为自动删除对象(如日志文件)。有关更多信息，请参阅 the section called “设置生命周期配置”。

有关 Amazon S3 特征和定价的信息，请参阅 Amazon S3 定价。

您可以使用 Amazon S3 控制台、AWS SDK 或 REST API 从 S3 存储桶中同时删除多个对象。

使用 S3 控制台

请按照以下步骤使用 Amazon S3 控制台从存储桶中删除多个对象。

> **Warning**
>
> - 删除指定的对象无法撤销。
> - 此操作将删除所有指定的对象。删除文件夹时，请等待删除操作完成，然后再将新对象添加到文件夹。否则，新对象也可能会被删除。
> - 当删除未启用版本控制的存储桶中的对象时，Amazon S3 将永久删除这些对象。

- 当删除已启用或已暂停存储桶版本控制的存储桶中的对象时，Amazon S3 将创建删除标记。有关更多信息，请参阅使用删除标记。

删除已启用或暂停版本控制的对象

> **Note**
>
> 如果已暂停版本控制的存储桶中对象的版本 ID 标记为 NULL，则 S3 会永久删除这些对象，因为不存在之前的版本。然而，如果针对已暂停版本控制的存储桶中的对象列出了有效的版本 ID，S3 会为已删除的对象创建删除标记，同时保留这些对象的先前版本。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶名称列表中，选择要从中删除对象的存储桶的名称。
3. 选择对象，然后选择删除。
4. 要确认删除指定的对象下的对象列表，请在删除对象？文本框中输入 **delete**。

永久删除已启用版本控制的存储桶中的特定对象版本

> **Warning**
>
> 当您永久删除 Amazon S3 中的特定对象版本时，删除操作无法撤销。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶名称列表中，选择要从中删除对象的存储桶的名称。
3. 选择您要删除的 对象。
4. 选择显示版本开关。
5. 选择对象版本，然后选择删除。
6. 要确认永久删除指定的对象下列出的特定对象版本，请在删除对象？文本框中输入 Permanently delete。Amazon S3 将永久删除特定对象版本。

永久删除未 启用版本控制的 Amazon S3 存储桶中的对象

> **Warning**
>
> 当您永久删除 Amazon S3 中的对象时，删除操作无法撤销。此外，对于任何未启用版本控制的桶，删除操作都是永久性的。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶名称列表中，选择要从中删除对象的存储桶的名称。
3. 选择对象，然后选择删除。
4. 要确认永久删除指定的对象下列出的对象，请在删除对象？文本框中输入 permanently delete。

> **Note**
>
> 如果您在删除对象时遇到任何问题，请参阅我想永久删除受版本控制的对象。

使用 AWS 软件开发工具包

有关如何使用 AWS SDK 删除多个对象的示例，请参阅将 DeleteObjects 与 AWS SDK 或 CLI 配合使用。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

使用 REST API

您可以使用 AWS SDK，通过多个对象删除 API 删除多个对象。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。

有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的删除多个对象。

组织、列出和处理对象

在 Amazon S3 中，您可以使用前缀来组织存储。前缀是存储桶中对象的逻辑分组。前缀值类似于允许您将相似数据存储在存储桶内同一目录下的目录名称。以编程方式上传对象时，可以使用前缀来组织数据。

在 Amazon S3 控制台中，前缀称为文件夹。您可以通过导航到存储桶，在 S3 控制台中查看所有对象和文件夹。您还可以查看有关每个对象的信息，包括对象属性。

有关在 Amazon S3 中列出和组织数据的更多信息，请参阅以下主题。

主题

使用前缀组织对象

您可以使用前缀来组织存储在 Amazon S3 存储桶中的数据。前缀是对象键名称开头的一串字符串。前缀可以是任意长度，取决于对象键名称的最大长度（1024 个字节）。您可以把前缀视为一种以类似于目录的方式组织数据的方式。但是，前缀不是目录。

按前缀搜索会将结果限制为仅以指定前缀开头的这些键。分隔符导致列表操作将共享公共前缀的所有键汇总到单个汇总列表结果中。

前缀和分隔符参数的目的是帮助您按层次结构组织，然后浏览您的键。要执行此操作，首先为您的存储桶选取一个分隔符，例如，斜杠（/），它不会出现在任何预期的键名中。您可以使用另一个字符作为分隔符。斜杠 (/) 字符没有什么独特之处，它是一个非常常见的前缀分隔符。接下来，通过串联所有包含层次结构的级别，并使用分隔符分隔每个级别来构建您的键名。

例如，如果您正在存储关于城市的信息，您可以按大陆、按国家/区域，然后按省份或州来自然地组织他们。因为这些名称通常不包含标点符号，您可以使用斜杠 (/) 作为分隔符。下面的示例使用斜杠 (/) 分隔符。

- Europe/France/Nouvelle-Aquitaine/Bordeaux
- North America/Canada/Quebec/Montreal
- North America/USA/Washington/Bellevue
- North America/USA/Washington/Seattle

如果您采用此方式来存储世界上每个城市的数据，那么管理平面键命名空间会很困难。通过使用带列表操作的 `Prefix` 和 `Delimiter`，您可以使用已创建的层次结构来列出您的数据。例如，要列出美国的

所有州，请设置 Delimiter='/' 和 Prefix='North America/USA/'。要列出您拥有数据的所有加拿大省份，请设置 Delimiter='/' 和 Prefix='North America/Canada/'。

有关分隔符、前缀和嵌套文件夹的更多信息，请参阅前缀和嵌套文件夹之间的区别。

使用前缀和分隔符列出对象

如果您使用分隔符发布列表请求，您可以仅浏览一个级别的层次结构、跳过或总结嵌套在更深级别的键（可能是数百万）。例如，假设您拥有具有以下键的存储桶（*DOC-EXAMPLE-BUCKET*）：

sample.jpg

photos/2006/January/sample.jpg

photos/2006/February/sample2.jpg

photos/2006/February/sample3.jpg

photos/2006/February/sample4.jpg

示例存储桶仅拥有根级 sample.jpg 对象。要仅列出存储桶中的根级对象，您需要向存储桶发送带正斜杠（/）分隔符的 GET 请求。作为响应，Amazon S3 将返回 sample.jpg 对象密钥，因为它不包含 / 分隔符。所有其他键都包含分隔符。Amazon S3 将组合这些密钥，并在指定的分隔符首次出现时，返回带前缀值 photos/ 的单个 CommonPrefixes 元素（它是这些密钥开头的子字符串）。

Example

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>DOC-EXAMPLE-BUCKET</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>sample.jpg</Key>
    <LastModified>2011-07-24T19:39:30.000Z</LastModified>
    <ETag>&quot;d1a7fb5eab1c16cb4f7cf341cf188c3d&quot;</ETag>
    <Size>6</Size>
    <Owner>
      <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>displayname</DisplayName>
    </Owner>
```

```
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>photos/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

有关以编程方式列出对象键的更多信息，请参阅以编程方式列出对象键。

以编程方式列出对象键

在 Amazon S3 中，键可以按前缀列出。您可以为相关键的名称选择通用前缀，然后使用分隔层次结构的特殊字符标记这些键。然后，您可以使用列表操作按层次选择和浏览键。这类似于在文件系统的目录中存储文件的方式。

Amazon S3 公开了列表操作，允许您列出包含在存储桶中的键。将按存储桶和前缀选择用于列表的键。例如，假设一个存储桶的名称为“dictionary”，它为每个英语词汇包含了一个键值。您可能会创建一个调用来列出该存储桶中以字母“q”开头的存储桶。列表结果始终以 UTF-8 二进制顺序返回。

SOAP 和 REST 列表操作将返回一个 XML 文档，其中包含匹配键值的名称和有关由每个键值识别的对象的信息。

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 特征。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

出于列表目的，可以根据通用前缀收拢那些共享一个前缀并且以特定分隔符终结的键组。这允许应用程序按层次结构组织和浏览键值，与您在文件系统的目录中组织文件的方式相同。

例如，要扩展字典存储桶以包含除英语单词外的更多内容，您可以使用语言和分隔符为每个单词添加前缀（例如，“French/logical”），从而构成键值。使用此命名方案和层级列表特征，您可以检索仅包含法语词汇的列表。您也可以浏览可用语言的顶级列表，而无需循环浏览所有按字典顺序排列的干预键。有关列表的此方面的更多信息，请参阅 使用前缀组织对象。

REST API

如果您的应用程序需要它，则可以直接发送 REST 请求。您可以发送 GET 请求来返回存储桶中的某些或所有对象，或者您也可以使用选择条件来返回存储桶中对象的子集。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 GET Bucket (List Objects) 版本 2。

列表执行效率

存储桶中的密钥总数不会对列表性能产生重大影响。它也不因是否存在 `prefix`、`marker`、`maxkeys` 或 `delimiter` 参数而受到影响。

循环访问多页结果

由于存储桶可以包含几乎无限数量的键，列表查询的完整结果可能会非常大。为了管理大型结果集，Amazon S3 API 支持分页，以将它们分割为多个响应。每个列出键响应将返回一个拥有多达 1000 个键的页面，同时使用指示器来指示响应是否存在截断。您可以发送一系列的列出键请求，直到您收到了所有的键。AWSSDK 包装库提供相同的分页。

示例

以下代码示例演示如何使用 `ListObjects`。

CLI

AWS CLI

以下示例使用 `list-objects` 命令显示指定存储桶中所有对象的名称：

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

该示例使用 `--query` 参数筛选 `list-objects` 的输出，使其范围缩小到每个对象的键值和大小

有关对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“使用 Amazon S3 对象”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListObjects。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令检索有关存储桶“test-files”中所有项目的信息。

```
Get-S3Object -BucketName test-files
```

示例 2：此命令从存储桶“test-files”中检索有关项目“sample.txt”的信息。

```
Get-S3Object -BucketName test-files -Key sample.txt
```

示例 3：此命令从存储桶“test-files”中检索有关前缀为“sample”的所有项目的信息。

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 ListObjects。

使用文件夹在 Amazon S3 控制台中整理对象

在 Amazon S3 中，存储桶和对象是主要资源，并且对象存储在存储桶中。Amazon S3 具有扁平结构，而不是类似于您在文件系统中看到的层次结构。不过，为了实现组织简易性，Amazon S3 控制台支持将文件夹概念作为对象分组手段。控制台通过为分组对象使用共享名称前缀来实现此目的。换句话说，分组对象的名称以一个公用字符串开头。这个公用字符串或共享前缀是文件夹名称。对象名称也称为键名称。

例如，您可以在控制台中创建名为 photos 的文件夹，并在其中存储名为 myphoto.jpg 的对象。随后，将使用键名 photos/myphoto.jpg 存储对象，其中 photos/ 为前缀。

以下是另外两个示例：

- 如果您的存储桶中有三个对象：logs/date1.txt、logs/date2.txt 和 logs/date3.txt，则控制台会显示名为 logs 的文件夹。如果您在控制台中打开该文件夹，将看到三个对象：date1.txt、date2.txt 和 date3.txt。
- 如果您有名为 photos/2017/example.jpg 的对象，则控制台会显示名为 photos 的文件夹，其中包含文件夹 2017。文件夹 2017 将包含对象 example.jpg。

文件夹中可以有文件夹，但存储桶中不能有存储桶。可以直接将对象上传和复制到一个文件夹中。可以创建和删除文件夹以及将文件夹设为公用，但不能对文件夹进行重命名。可以将对象从一个文件夹复制至另一个文件夹。

> **Important**
>
> 当您在 Amazon S3 中创建文件夹时，S3 会使用设置为您提供的文件夹名称的密钥创建一个 0 字节对象。例如，如果在存储桶中创建一个名为 photos 的文件夹，Amazon S3 控制台将使用键 photos/ 创建一个 0 字节的对象。控制台创建这个对象是为了支持文件夹的概念。对于将正斜杠 (/) 字符作为键名称中的最后一个（尾部）字符的所有对象（例如 examplekeyname/），Amazon S3 控制台将其视为文件夹。您无法使用 Amazon S3 控制台

上传键名称中有尾部 / 字符的对象。但是，通过使用 AWS Command Line Interface（AWS CLI）、AWS SDK 或 REST API，您可以利用 Amazon S3 API 上传名称中有尾部 / 的对象。名称中有尾部 / 的对象显示为 Amazon S3 控制台中的文件夹。Amazon S3 控制台不为此类对象显示内容和元数据。当使用控制台复制名称中有尾部 / 的对象时，将在目标位置创建一个新文件夹，但不会复制对象的数据和元数据。

主题

创建文件夹

本部分介绍如何使用 Amazon S3 控制台创建文件夹。

Important

如果存储桶策略阻止在没有标签、元数据或访问控制列表（ACL）被授权者的情况下将对象上传到此存储桶，则您无法使用以下过程创建文件夹。而应改为上传空文件夹，并在上传配置中指定以下设置。

如何创建文件夹

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（存储桶）列表中，请选择要在其中创建文件夹的存储桶的名称。
4. 如果存储桶策略禁止在不加密的情况下将对象上传到此存储桶，则必须在 Server-side encryption（服务器端加密）下选择 Enable（启用）。
5. 请选择 Create folder（创建文件夹）。
6. 输入文件夹的名称（例如，**`favorite-pics`**）。然后选择 Create folder（创建文件夹）。

将文件夹设为公用

我们建议禁止所有对 Amazon S3 文件夹和存储桶的公有访问，除非您特别需要公有文件夹或存储桶。当您将文件夹设为公有时，Internet 上的任何人都可以查看该文件夹中分组的所有对象。

在 Amazon S3 控制台中，您可以将文件夹设为公有。您还可以通过创建存储桶策略来将文件夹设为公有，该策略通过前缀限制数据访问权限。有关更多信息，请参阅 Amazon S3 的身份和访问管理。

> **Warning**
>
> 在 Amazon S3 控制台中将文件夹设为公有后，就不能再将其设为私有。而是必须对公有文件夹中的每个单独的对象设置权限，以使对象不具备公有访问。有关更多信息，请参阅 配置 ACL。

主题

计算文件夹大小

本节介绍如何使用 Amazon S3 控制台计算文件夹的大小。

计算文件夹的大小

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（存储桶）列表中，选择在其中存储文件夹的存储桶的名称。
4. 在 Objects（对象）列表中，选中文件夹名称旁边的复选框。
5. 选择 Actions（操作），然后选择 Calculate total size（计算总大小）。

> **Note**
>
> 在您导航离开该页面后，文件夹信息（包括总大小）将不再可用。如果您想再次看到总大小，则必须再次计算该值。

Important

当您对存储桶中的指定对象或文件夹使用 Calculate total size（计算总大小）操作时，Amazon S3 会计算对象总数和总存储大小。但是，在对象总数或总大小中并不计算未完成或正在进行的分段上传以及以前或非当前的版本。此操作仅针对存储在存储桶中的每个对象的当前或最新版本计算对象总数和总大小。

例如，如果存储桶中有某个对象的两个版本，则 Amazon S3 中的存储计算器仅将它们计为一个对象。因此，在 Amazon S3 控制台中计算的对象总数可能不同于 S3 Storage Lens 存储统计管理工具中显示的 Object Count（对象计数）指标以及 Amazon CloudWatch 指标 `NumberOfObjects` 报告的数量。同样，总存储大小也可能不同于 S3 Storage Lens 存储统计管理工具中显示的 Total Storage（总存储）指标和 CloudWatch 中显示的 `BucketSizeBytes` 指标。

删除文件夹

本节介绍如何使用 Amazon S3 控制台从 S3 存储桶中删除文件夹。

有关 Amazon S3 特征和定价的信息，请参阅 Amazon S3。

要从 S3 存储桶中删除文件夹

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要从中删除文件夹的存储桶的名称。
3. 在 Objects（对象）列表中，选中您要删除的文件夹和对象旁的复选框。
4. 请选择 Delete。
5. 在 Delete objects (删除对象) 页面上，验证是否已列出选择删除的文件夹的名称。
6. 在 Delete objects (删除对象) 框中，输入 **`delete`**，然后选择 Delete objects (删除对象)。

Warning

此操作将删除所有指定的对象。删除文件夹时，请等待删除操作完成，然后再将新对象添加到文件夹。否则，新对象也可能会被删除。

在 Amazon S3 控制台中查看对象概述

您可以使用 Amazon S3 控制台查看对象的概述。控制台在一个位置提供对象的所有基本信息。

打开对象的详细信息页面

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket（存储桶）列表中，请选择包含对象的存储桶的名称。
3. 在 Objects（对象）列表中，请选择您希望提供概述的对象的名称。

   将打开对象详细信息页面。
4. 要下载对象，请选择 Object actions（对象操作），然后选择 Download（下载）。要将对象的路径复制到剪贴板，请在 Object URL（对象 URL）下选择 URL。
5. 如果对存储桶启用了版本控制，请选择 Versions（版本）以列出对象的版本。
   - 要下载对象版本，请选中版本 ID 旁边的复选框，请选择 Actions（操作），然后选择 Download（下载）。
   - 要删除对象版本，请选中版本 ID 旁边的复选框，然后选择 Delete（删除）。

> **Important**
>
> 仅当对象已作为最新 (当前) 版本删除时，您才能取消删除它。您无法取消删除已删除对象的早期版本。

在 Amazon S3 控制台中查看对象属性

您可以使用 Amazon S3 控制台查看对象的属性，包括存储类、加密设置、标签和元数据。

查看对象的属性

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket（存储桶）列表中，请选择包含对象的存储桶的名称。
3. 在 Objects（对象）列表中，请选择要查看其属性的对象的名称。

对象的 Object overview（对象概述）将打开。您可以向下滚动以查看对象属性。

4. 在 Object overview（对象概述）页面上，您可以为对象配置以下属性。

> **Note**
>
> - 如果您更改存储类、加密或元数据属性，则将创建一个新对象来替换旧对象。如果启用 S3 版本控制，则会创建对象的新版本，而现有对象将变为旧版本。更改属性的角色也会成为新对象或（对象版本）的拥有者。
> - 如果您要更改具有用户定义标签的对象的存储类、加密或元数据属性，您必须拥有 `s3:GetObjectTagging` 权限。如果您要对没有用户定义标签但大小超过 16 MB 的对象更改这些属性，您还必须拥有 `s3:GetObjectTagging` 权限。
>
>   如果目标存储桶策略拒绝 `s3:GetObjectTagging` 操作，则将更新对象的这些属性，但将从对象中移除用户定义的标签，并且您将收到错误。

a. Storage class (存储类) – Amazon S3 中的每个对象都有与之关联的存储类。您选择使用的存储类取决于您访问对象的频率。S3 对象的默认存储类是 STANDARD。您可以选择在上传对象使用的存储类。有关存储类的更多信息，请参阅 使用 Amazon S3 存储类。

   要在上传对象后更改存储类，请选择 Storage class (存储类)。请选择您希望的存储类，然后选择 Save（保存）。

b. 服务器端加密设置 – 您可以使用服务器端加密来加密 S3 对象。有关更多信息，请参阅使用 AWS KMS (SSE-KMS) 指定服务器端加密或指定具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

c. 元数据 – Amazon S3 中的每个对象都有一组表示其元数据的名称/值对。有关将元数据添加到 S3 对象的信息，请参阅在 Amazon S3 控制台中编辑对象元数据。

d. 标签 – 您可以通过向 S3 对象添加标签来对存储进行分类。有关更多信息，请参阅 使用标签对存储进行分类。

e. 对象锁定依法保留和保留 – 您可以防止对象被删除。有关更多信息，请参阅 使用 S3 对象锁定。

使用预签名 URL

您可以使用预签名 URL 授予对 Amazon S3 中对象的限时访问权限，而不更新存储桶策略。可以在浏览器中输入预签名 URL，或者程序使用预签名 URL 来下载对象。预签名 URL 使用的凭证是生成该 URL 的 AWS 用户的凭证。

还可以使用预签名 URL 来允许他人将特定对象上传到您的 Amazon S3 存储桶。这允许在不要求另一方拥有 AWS 安全凭证或权限的情况下进行上传。如果具有相同键的对象已存在于在预签名 URL 中指定的存储桶中，则 Amazon S3 将现有对象替换为上传的对象。

在到期日期和时间之前，可以多次使用预签名 URL。

创建预签名 URL 时，必须提供您的安全凭证，然后指定以下内容：

- 一个 Amazon S3 存储桶
- 对象键（如果将在您的 Amazon S3 存储桶中下载此对象，则一旦上传，这就是要上传的文件名）
- HTTP 方法（GET 用于下载对象，或 PUT 用于上传）
- 过期时间间隔

目前，Amazon S3 预签名 URL 不支持在上传对象时使用以下数据完整性校验和算法（CRC32、CRC32C、SHA-1、SHA-256）。要在上传后验证对象的完整性，您可以在使用预签名 URL 上传对象时提供对象的 MD5 摘要。有关对象完整性的更多信息，请参阅检查对象完整性。

主题

谁可以创建预签名 URL

具有有效安全凭证的任何人都可以创建预签名 URL。但对于成功地访问对象的人来说，必须由拥有执行预签名 URL 所基于的操作权限的人创建预签名 URL。

以下各个类型的凭证可用于创建预签名 URL：

- IAM 实例配置文件 – 有效期最长 6 小时。
- AWS Security Token Service – 有效期为最长 36 小时（使用长期安全凭证进行签名时），或为临时凭证的持续时间，以先到者为准。
- IAM 用户 – 使用 AWS 签名版本 4 时，有效期最长 7 天。

  要创建有效期最长为 7 天的预签名 URL，请首先将 IAM 用户凭证（访问密钥和私有密钥）委托给用于创建预签名 URL 的方法。

> **Note**
>
> 如果您使用临时凭证创建了预签名 URL，则此 URL 将在凭证过期时过期。通常，当您用于创建预签名 URL 的凭证被撤销、删除或停用时，预签名 URL 就会到期。即使创建的 URL 的过期时间更晚，也是如此。有关临时安全凭证生命周期，请参阅《IAM 用户指南》中的比较 AWS STS API 操作。

预签名 URL 的到期时间

预签名 URL 在生成 URL 时指定的时间段内保持有效。如果使用 Amazon S3 控制台创建预签名 URL，过期时间可设置为 1 分钟到 12 小时之间。如果您使用 AWS CLI 或 AWS SDK，则过期时间可设置为多达 7 天。

如果您使用临时令牌创建了预签名 URL，则此 URL 将在该令牌到期时到期。通常，当您用于创建预签名 URL 的凭证被撤销、删除或停用时，预签名 URL 就会到期。即使创建的 URL 的过期时间更晚，也是如此。有关您使用的凭证如何影响到期时间的更多信息，请参阅谁可以创建预签名 URL。

在发出 HTTP 请求时，Simple Storage Service (Amazon S3) 会检查签名 URL 的到期日期和时间。例如，如果客户端刚好在到期时间之前开始下载某个大型文件，即使在下载过程中超过到期时间，下载也会继续进行。但如果连接断开，在客户端试图在超过到期时间后重新开始下载，则下载将会失败。

限制预签名 URL 功能

预签名 URL 的功能受创建它的用户的权限所限制。预签名 URL 实质上是一种不记名令牌，向持有相关 URL 的人授予了访问权限。因此，我们建议您适当地保护它们。以下是一些可用来限制使用预签名 URL 的方法。

AWS 签名版本 4（SigV4）

使用 AWS 签名版本 4（SigV4）对预签名 URL 请求进行身份认证时要强制执行特定的行为，您可以在存储桶策略和接入点策略中使用条件键。例如，以下存储桶策略使用 s3:signatureAge 条件，当相关签名的存在时间超过 10 分钟时，拒绝对于 *amzn-s3-demo-bucket1* 存储桶中对象的任何 Amazon S3 预签名 URL 请求。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Deny a presigned URL request if the signature is more than 10 min
 old",
         "Effect": "Deny",
         "Principal": {"AWS":"*"},
         "Action": "s3:*",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
         "Condition": {
            "NumericGreaterThan": {
               "s3:signatureAge": 600000
            }
         }
      }
   ]
}
```

有关策略键相关的 AWS 签名版本 4 的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 AWS 签名版本 4 身份认证。

网络路径限制

如果您想限制使用预签名 URL 和 Amazon S3 对特定网络路径的所有访问权限，则可以编写 AWS Identity and Access Management（IAM）策略。您可以在进行调用的 IAM 主体、Amazon S3 存储桶或两者上设置这些策略。

对 IAM 主体实施网络路径限制后，要求拥有这些凭证的用户从指定的网络发出请求。对存储桶或接入点实施限制后，要求对该资源的所有请求都必须来自指定的网络。这些限制也适用于预签名 URL 以外的场景。

您使用的 IAM 全局条件键取决于端点的类型。如果您正在对 Amazon S3 使用公有端点，请使用 aws:SourceIp。如果您正在使用虚拟私有云（VPC）端点访问 Amazon S3 存储桶，请使用 aws:SourceVpc 或 aws:SourceVpce。

以下 IAM 策略语句要求主体仅从指定的网络范围访问。AWS使用此策略语句时，所有访问都必须来自该范围。这包括有人对于 Amazon S3 使用预签名 URL 的情况。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

使用预签名 URL 共享对象

默认情况下，所有 Amazon S3 对象都是私有的，只有对象拥有者才具有访问它们的权限。但是，对象拥有者可以通过创建预签名 URL 与其他人共享对象。预签名 URL 使用安全凭证来授予下载对象的限时权限。可以在浏览器中输入此 URL，或者程序使用此 URL 来下载对象。预签名 URL 使用的凭证是生成该 URL 的 AWS 用户的凭证。

有关预签名 URL 的一般信息，请参阅使用预签名 URL。

您可以使用 Amazon S3 控制台、适用于 Visual Studio 的 AWS Explorer 或 AWS Toolkit for Visual Studio Code 创建预签名 URL 来共享对象，而不需要编写任何代码。也可以使用 AWS Command Line Interface（AWS CLI）或 AWS SDK 以编程方式生成预签名 URL。

使用 S3 控制台

您可以使用 Amazon S3 控制台，按照以下步骤生成预签名 URL 来共享对象。使用控制台时，预签名 URL 的最长过期时间为自创建时起 12 小时。

使用 Amazon S3 控制台生成预签名 URL

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（存储桶）列表中，请选择包含要为其生成预签名 URL 的对象的存储桶的名称。

4. 在 Objects（对象）列表中，选择要为其生成预签名 URL 的对象。
5. 在对象操作菜单上，请选择使用预签名 URL 共享。
6. 指定您希望的预签名 URL 有效时间长度。
7. 请选择 Create presigned URL（创建预签名 URL）。
8. 出现确认时，URL 将自动复制到剪贴板。如果您需要再次复制预签名 URL，您将看到一个按钮，用于复制该 URL。

使用 AWS CLI

以下示例 AWS CLI 命令生成一个预签名 URL，以共享 Amazon S3 存储桶中的对象。使用 AWS CLI 时，预签名 URL 的最长过期时间为自创建时起 7 天。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3 presign s3://amzn-s3-demo-bucket1/mydoc.txt --expires-in 604800
```

> **Note**
>
> 对于 2019 年 3 月 20 日之后启动的所有 AWS 区域，您需要随请求指定 endpoint-url 和 AWS ##。有关所有 Amazon S3 区域和端点的列表，请参阅《AWS 一般参考》中的区域和端点。

```
aws s3 presign s3://amzn-s3-demo-bucket1/mydoc.txt --expires-in 604800 --region af-south-1 --endpoint-url https://s3.af-south-1.amazonaws.com
```

有关更多信息，请参阅 AWS CLI 命令参考 中的 presign。

使用 AWS SDK

有关使用 AWS SDK 生成预签名 URL 来共享对象的示例，请参阅使用 AWS SDK 为 Amazon S3 创建预签名 URL。

使用 AWS SDK 生成预签名 URL 时，最长到期时间为创建之时起 7 天。

> **Note**
>
> 对于 2019 年 3 月 20 日之后启动的所有 AWS 区域，您需要随请求指定 `endpoint-url` 和 `AWS ##`。有关所有 Amazon S3 区域和端点的列表，请参阅《AWS 一般参考》中的区域和端点。

> **Note**
>
> 使用 AWS SDK 时，Tagging 属性必须是标题而不是查询参数。所有其他属性都可以作为预签名 URL 的参数传递。

使用 AWS Toolkit for Visual Studio (Windows)

> **Note**
>
> 目前，AWS Toolkit for Visual Studio 不支持 Visual Studio for Mac。

1. 按照《AWS Toolkit for Visual Studio User Guide》中的 Installing and setting up the Toolkit for Visual Studio 中的说明安装 AWS Toolkit for Visual Studio。
2. 使用以下步骤（《AWS Toolkit for Visual Studio 用户指南》中的连接到 AWS）连接到 AWS。
3. 在标有 AWS 各区服务浏览器的左侧面板中，双击包含您的对象的存储桶。
4. 右键单击要为其生成预签名 URL 的对象，然后选择创建预签名 URL...。
5. 在弹出窗口中，设置预签名 URL 的到期日期和时间。
6. 应根据您选择的对象预填充对象密钥。
7. 选择 GET 可指定此预签名 URL 将用于下载对象。
8. 选择 Generate（生成）按钮。
9. 要将此 URL 复制到剪贴板，请选择 Copy（复制）。
10. 要使用生成的预签名 URL，请将该 URL 粘贴到任何浏览器中。

使用 AWS Toolkit for Visual Studio Code

如果您使用的是 Visual Studio 代码，可以使用 AWS Toolkit for Visual Studio Code 生成预签名 URL 来共享对象的，而不需要编写任何代码。有关一般信息，请参阅《AWS Toolkit for Visual Studio Code 用户指南》中的 AWS Toolkit for Visual Studio Code。

有关如何安装 AWS Toolkit for Visual Studio Code 的说明，请参阅《AWS Toolkit for Visual Studio Code 用户指南》中的安装 AWS Toolkit for Visual Studio Code。

1. 使用以下步骤（《AWS Toolkit for Visual Studio Code 用户指南》中的连接到 AWS Toolkit for Visual Studio Code）连接到 AWS。
2. 在 Visual Studio 代码中选择左侧面板上的 AWS 徽标。
3. 在资源管理器下，选择 S3。
4. 选择存储桶和文件，然后打开上下文菜单（右键单击）。
5. 选择生成预签名 URL，然后设置过期时间（以分钟为单位）。
6. 按 Enter，预签名 URL 将复制到剪贴板。

使用预签名 URL 上传对象

您可以使用预签名 URL 来允许他人将对象上传到您的 Amazon S3 存储桶。使用预签名 URL 将允许在不要求另一方拥有 AWS 安全凭证或权限的情况下进行上传。预签名 URL 受创建它的用户的权限所限制。也即，如果您收到预签名 URL 来上传对象，则仅当该 URL 的创建者拥有上传该对象所需的权限时，您才能上传对象。

当有人使用该 URL 上传对象时，Amazon S3 将在指定的存储桶中创建对象。如果存储桶中已存在具有预签名 URL 中指定的相同键的对象，则 Amazon S3 会将现有对象替换为上传的对象。上传后，存储桶拥有者将拥有该对象。

有关预签名 URL 的一般信息，请参阅使用预签名 URL。

可以使用适用于 Visual Studio 的 AWS Explorer 生成预签名 URL 来上传对象，而不需要编写任何代码。也可以使用 AWS SDK 以编程方式生成预签名 URL。

使用 AWS Toolkit for Visual Studio (Windows)

> **Note**
>
> 目前，AWS Toolkit for Visual Studio 不支持 Visual Studio for Mac。

1. 按照《AWS Toolkit for Visual Studio User Guide》中的 Installing and setting up the Toolkit for Visual Studio 中的说明安装 AWS Toolkit for Visual Studio。
2. 使用以下步骤（《AWS Toolkit for Visual Studio 用户指南》中的连接到 AWS）连接到 AWS。
3. 在标有 AWS 各区服务浏览器的左侧面板中，右键单击要向其中上传对象的存储桶。
4. 选择创建预签名 URL...。
5. 在弹出窗口中，设置预签名 URL 的到期日期和时间。
6. 在对象键中，设置要上传的文件的名称。您上传的文件必须与该名称完全匹配。如果具有相同对象键的对象已位于存储桶中，Amazon S3 会将现有对象替换为新上传的对象。
7. 选择 PUT 可指定此预签名 URL 将用于上传对象。
8. 选择 Generate（生成）按钮。
9. 要将此 URL 复制到剪贴板，请选择 Copy（复制）。
10. 要使用此 URL，您可以通过 `curl` 命令发送 PUT 请求。包括您的文件的完整路径以及预签名 URL 本身。

```
curl -X PUT -T "/path/to/file" "presigned URL"
```

使用 AWS SDK

有关使用 AWS SDK 生成预签名 URL 来上传对象的示例，请参阅使用 AWS SDK 为 Amazon S3 创建预签名 URL。

使用 AWS SDK 生成预签名 URL 时，最长到期时间为创建之时起 7 天。

> **Note**
>
> 对于 2019 年 3 月 20 日之后启动的所有 AWS 区域，您需要随请求指定 `endpoint-url` 和 `AWS ##`。有关所有 Amazon S3 区域和端点的列表，请参阅《AWS 一般参考》中的区域和端点。

使用 S3 对象 Lambda 转换对象

借助 Amazon S3 对象 Lambda，您可以将自己的代码添加到 Amazon S3 `GET`、`LIST` 和 `HEAD` 请求中，以便在数据返回到应用程序时修改和处理数据。您可以使用自定义代码修改由 S3 `GET` 请求返回的数据，以便执行筛选行、动态调整图像大小和给图像加水印、隐去机密数据等操作。您还可以

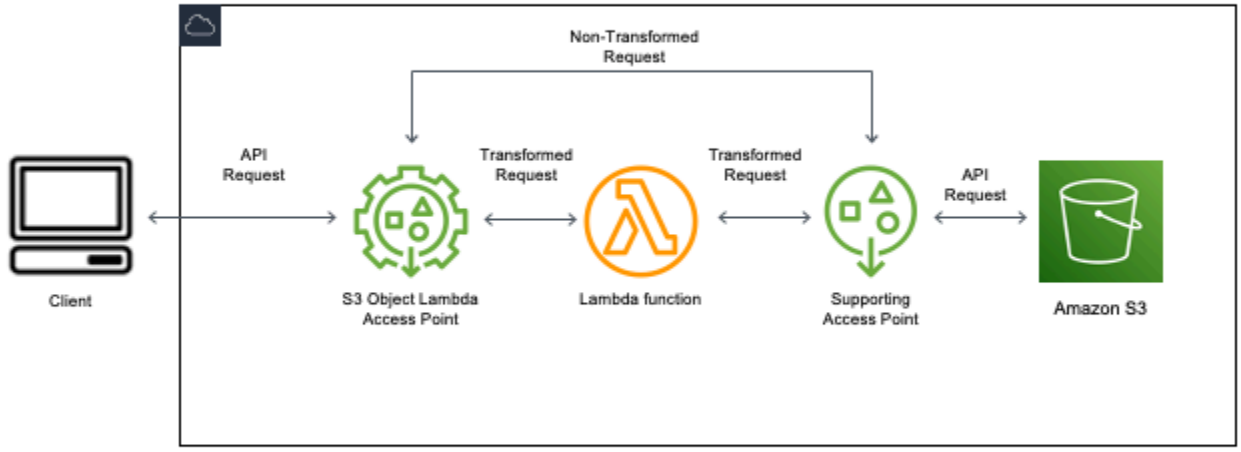
使用 S3 对象 Lambda 修改 S3 LIST 请求的输出以创建存储桶中所有对象的自定义视图，以及修改 S3 HEAD 请求以修改对象元数据（如对象名称和大小）。您可以使用 S3 对象 Lambda 作为 Amazon CloudFront 分配的来源，以便为最终用户量身定制数据，例如自动调整图像大小、对旧格式进行转码（例如从 JPEG 转码为 WebP）或剥离元数据。有关更多信息，请参阅 AWS 博客文章将 Amazon S3 对象 Lambda 与 Amazon CloudFront 结合使用。在 AWS Lambda 函数的支持下，您的代码在完全由 AWS 管理的基础设施上运行。使用 S3 对象 Lambda 可以减少创建和存储数据的衍生副本或运行代理的需要，所有这些都无需更改应用程序。

S3 对象 Lambda 的工作原理

S3 对象 Lambda 使用 AWS Lambda 函数来自动处理标准 S3 GET、LIST 或 HEAD 请求的输出。AWS Lambda 是一种无服务器计算服务，它运行客户定义的代码，而无需管理底层计算资源。您可以编写和运行自己的自定义 Lambda 函数，根据您的特定使用案例定制数据转换。

配置 Lambda 函数后，您可以将其附加到 S3 对象 Lambda 服务端点（称为对象 Lambda 接入点）。对象 Lambda 接入点使用标准 S3 接入点（称为支持接入点）来访问 Amazon S3。

当您向对象 Lambda 接入点发送请求时，Amazon S3 会自动调用您的 Lambda 函数。然后，使用 S3 GET、LIST 或 HEAD 请求通过对象 Lambda 接入点检索的任何数据都会将转换后的结果返回给应用程序。将正常处理所有其他请求，如下图所示。



本节中的主题介绍了如何使用 S3 对象 Lambda。

主题

创建对象 Lambda 接入点

对象 Lambda 接入点恰好与一个标准接入点相关联，因此也与一个 Amazon S3 存储桶相关联。要创建对象 Lambda 接入点，您需要以下资源：

- Amazon S3 存储桶。有关创建存储桶的更多信息，请参阅 the section called “创建存储桶”。
- 标准 S3 接入点。当您使用对象 Lambda 接入点时，此标准接入点称为支持接入点。有关创建标准接入点的信息，请参阅the section called “创建接入点”。
- AWS Lambda 函数。您可以创建自己的 Lambda 函数，也可以使用预构建的函数。有关创建 Lambda 函数的更多信息，请参阅the section called “编写 Lambda 函数”。有关预构建的函数的更多信息，请参阅 使用 AWS 构建的 Lambda 函数。
- （可选）AWS Identity and Access Management（IAM）策略。Amazon S3 接入点支持 IAM 资源策略，您可以使用这些策略按资源、用户或其他条件控制接入点的使用。有关创建这些策略的更多信息，请参阅the section called “配置 IAM 策略”。

以下各节介绍如何使用以下方法创建对象 Lambda 接入点：

- 这些区域有：AWS Management Console
- AWS Command Line Interface (AWS CLI)
- AWS CloudFormation 模板
- 这些区域有：AWS Cloud Development Kit (AWS CDK)

有关如何使用 REST API 创建对象 Lambda 接入点的信息，请参阅《Amazon Simple Storage Service API 参考》中的 CreateAccessPointForObjectLambda。

创建对象 Lambda 接入点

使用以下过程之一创建对象 Lambda 接入点。

使用 S3 控制台

使用控制台创建对象 Lambda 接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要切换到的区域。
3. 在左侧导航窗格中，选择 Object Lambda 接入点。
4. 在对象 Lambda 接入点页面上，选择创建对象 Lambda 接入点。
5. 对于对象 Lambda 接入点名称，输入要用于接入点的名称。

   与标准接入点一样，对于对象 Lambda 接入点同样具有命名规则。有关更多信息，请参阅 命名 Amazon S3 接入点的规则。
6. 对于 Supporting Access Point（支持接入点），请输入或浏览到要使用的标准接入点。接入点必须与要变换的对象位于同一 AWS 区域 位置。有关创建标准接入点的信息，请参阅the section called “创建接入点”。
7. 在转换配置下，您可以添加一个函数来转换对象 Lambda 接入点的数据。请执行以下操作之一：

   - 如果您的账户中已经有一个 AWS Lambda 函数，则可以在 Invoke Lambda function（调用 Lambda 函数）下选择该函数。在这里，您可以在您的 AWS 账户中输入 Lambda 函数的 Amazon 资源名称（ARN），也可以从下拉菜单中选择 Lambda 函数。
   - 如果您想使用 AWS 构建的函数，请在 AWS 构建的函数下选择函数名称，然后选择创建 Lambda 函数。这将带您进入 Lambda 控制台，您可以在其中将构建的函数部署到您的 AWS 账户中。有关构建的函数的更多信息，请参阅使用 AWS 构建的 Lambda 函数。

   在 S3 APIs（S3 API）下，选择一个或多个要调用的 API 操作。对于所选的每个 API，您必须指定要调用的 Lambda 函数。
8. （可选）在 Payload（有效负载）下，添加要提供给 Lambda 函数作为输入的 JSON 文本。您可以为调用同一 Lambda 函数的不同对象 Lambda 接入点配置具有不同参数的负载，从而扩展 Lambda 函数的灵活性。

> **Important**
>
> 使用对象 Lambda 接入点时，确保负载不包含任何机密信息。

9. （可选）对于 Range and part number（范围和分段编号），如果您要处理带有范围和分段编号标头的 GET 和 HEAD 请求，则必须启用此选项。启用此选项将确认您的 Lambda 函数可以识别和处理这些请求。有关范围标头和分段编号的更多信息，请参阅 使用 Range 和 partNumber 标头。
10. （可选）对于请求指标，请选择启用或禁用，以将 Amazon S3 监控添加到对象 Lambda 接入点。请求指标按标准 Amazon CloudWatch 费率计费。
11. （可选）在对象 Lambda 接入点策略下，设置资源策略。资源策略授予针对指定对象 Lambda 接入点的权限，并可以按资源、用户或其他条件控制接入点的使用。有关对象 Lambda 接入点资源策略的更多信息，请参阅为对象 Lambda 接入点配置 IAM 策略。
12. 在屏蔽此对象 Lambda 接入点的公共访问权限设置下，选择要应用的屏蔽公共访问权限设置。默认情况下，为新的对象 Lambda 接入点启用所有屏蔽公共访问权限设置，建议您将默认设置保留为启用状态。Amazon S3 当前不支持在创建对象 Lambda 接入点之后更改对象 Lambda 接入点的屏蔽公共访问权限设置。

    有关使用 Amazon S3 屏蔽公共访问权限的更多信息，请参阅管理接入点的公有访问。
13. 请选择创建对象 Lambda 接入点。

使用 AWS CLI

使用 AWS CloudFormation 模板创建对象 Lambda 接入点

> **Note**
>
> 要使用以下命令，请将 *user input placeholders* 替换为您自己的信息。

1. 在 S3 对象 Lambda 默认配置中下载 AWS Lambda 函数部署包 `s3objectlambda_deployment_package.zip`。
2. 运行以下 `put-object` 命令以将文件包上传到 Amazon S3 存储桶。

```
aws s3api put-object --bucket Amazon S3 bucket name --key
 s3objectlambda_deployment_package.zip --body release/
s3objectlambda_deployment_package.zip
```

3. 在 S3 对象 Lambda 默认配置中下载 AWS CloudFormation 模板 `s3objectlambda_defaultconfig.yaml`。
4. 运行以下 `deploy` 命令以将模板部署到您的 AWS 账户。

```
aws cloudformation deploy --template-file s3objectlambda_defaultconfig.yaml \
 --stack-name AWS CloudFormation stack name \
 --parameter-overrides ObjectLambdaAccessPointName=Object Lambda Access Point name
 \
  SupportingAccessPointName=Amazon S3 access point S3BucketName=Amazon S3 bucket \
  LambdaFunctionS3BucketName=Amazon S3 bucket containing your Lambda package \
  LambdaFunctionS3Key=Lambda object key LambdaFunctionS3ObjectVersion=Lambda object
 version \
  LambdaFunctionRuntime=Lambda function runtime --capabilities capability_IAM
```

您可以将此 AWS CloudFormation 模板配置为针对 GET、HEAD 和 LIST API 操作调用 Lambda。有关修改模板的默认配置的更多信息，请参阅the section called “使用 AWS CloudFormation 自动进行 S3 对象 Lambda 设置”。

使用 AWS CLI 创建对象 Lambda 接入点

> **Note**
>
> 要使用以下命令，请将 *user input placeholders* 替换为您自己的信息。

以下示例为账户 *111122223333* 中的存储桶 *amzn-s3-demo-bucket1* 创建了一个名为 *my-object-lambda-ap* 的对象 Lambda 接入点。此示例假定已创建了名为 *example-ap* 的标准接入点。有关创建标准接入点的信息，请参阅 the section called “创建接入点”。

此示例使用 AWS 预构建的函数 `decompress`。有关预构建的函数的更多信息，请参阅 the section called “使用 AWS 构建的函数”。

1. 创建存储桶。在此示例中，我们将使用 *amzn-s3-demo-bucket1*。有关创建存储桶的更多信息，请参阅 the section called “创建存储桶”。
2. 创建标准接入点并将其附加到存储桶。在此示例中，我们将使用 *example-ap*。有关创建标准接入点的信息，请参阅the section called “创建接入点”。
3. 请执行以下操作之一：

- 在您的账户中创建一个 Lambda 函数，用于转换 Amazon S3 对象。有关创建 Lambda 函数的更多信息，请参阅the section called “编写 Lambda 函数”。要将您的自定义函数与 AWS CLI 结合使用，请参阅《AWS Lambda 开发人员指南》中的将 Lambda 与 AWS CLI 结合使用。
- 使用 AWS 预构建的 Lambda 函数。有关预构建的函数的更多信息，请参阅 使用 AWS 构建的 Lambda 函数。

4. 创建一个名为 `my-olap-configuration.json` 的 JSON 配置文件。在此配置中，为您在前面的步骤中创建的 Lambda 函数提供支持接入点和 Amazon 资源名称（ARN），或者为正在使用的预构建函数提供 ARN。

   Example

```
{
    "SupportingAccessPoint" : "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
    "TransformationConfigurations": [{
        "Actions" : ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation" : {
            "AwsLambda": {
                "FunctionPayload" : "{\"compressionType\":\"gzip\"}",
                "FunctionArn" : "arn:aws:lambda:us-east-1:111122223333:function/
compress"
            }
        }
    }]
}
```

5. 运行 `create-access-point-for-object-lambda` 命令以创建对象 Lambda 接入点。

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-ap --configuration file://my-olap-configuration.json
```

6. （可选）创建名为的 `my-olap-policy.json` JSON 策略文件。

   添加对象 Lambda 接入点资源策略可以按资源、用户或其他条件控制接入点的使用。此资源策略为账户 *444455556666* 授予针对指定对象 Lambda 接入点的 `GetObject` 权限。

   Example

```
{
```

```
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "Grant account 444455556666 GetObject access",
            "Effect": "Allow",
            "Action": "s3-object-lambda:GetObject",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Resource": "your-object-lambda-access-point-arn"
        }
    ]
}
```

7. （可选）运行 put-access-point-policy-for-object-lambda 命令以设置资源策略。

```
aws s3control put-access-point-policy-for-object-lambda --account-id 111122223333
 --name my-object-lambda-ap --policy file://my-olap-policy.json
```

8. （可选）指定负载。

有效负载是可选 JSON，您可以将其作为输入提供给 AWS Lambda 函数。您可以为调用同一 Lambda 函数的不同对象 Lambda 接入点配置具有不同参数的负载，从而扩展 Lambda 函数的灵活性。

以下对象 Lambda 接入点配置显示了带有两个参数的负载。

```
{
 "SupportingAccessPoint": "AccessPointArn",
 "CloudWatchMetricsEnabled": false,
 "TransformationConfigurations": [{
  "Actions": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
  "ContentTransformation": {
   "AwsLambda": {
    "FunctionArn": "FunctionArn",
    "FunctionPayload": "{\"res-x\": \"100\",\"res-y\": \"100\"}"
   }
  }
 }]
}
```

以下对象 Lambda 接入点配置显示具有一个参数以及启用了 GetObject-Range、GetObject-PartNumber、HeadObject-Range 和 HeadObject-PartNumber 的负载。

```
{
    "SupportingAccessPoint":"AccessPointArn",
    "CloudWatchMetricsEnabled": false,
    "AllowedFeatures": ["GetObject-Range", "GetObject-PartNumber", "HeadObject-
Range", "HeadObject-PartNumber"],
    "TransformationConfigurations": [{
        "Action": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation": {
            "AwsLambda": {
                "FunctionArn":"FunctionArn",
                "FunctionPayload": "{\"compression-amount\": \"5\"}"
            }
        }
    }]
}
```

> **Important**
>
> 使用对象 Lambda 接入点时，确保负载不包含任何机密信息。

使用 AWS CloudFormation 控制台和模板

您可以使用 Amazon S3 提供的默认配置创建对象 Lambda 接入点。您可以从 GitHub 存储库下载 AWS CloudFormation 模板和 Lambda 函数源代码，并部署这些资源以设置函数对象 Lambda 接入点。

有关修改 AWS CloudFormation 模板的默认配置的信息，请参阅the section called “使用 AWS CloudFormation 自动进行 S3 对象 Lambda 设置”。

有关在没有模板的情况下使用 AWS CloudFormation 配置对象 Lambda 接入点的信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::S3ObjectLambda::AccessPoint。

要上传 Lambda 函数部署软件包

1. 在 S3 对象 Lambda 默认配置中下载 AWS Lambda 函数部署包 s3objectlambda_deployment_package.zip。

2. 将文件包上传到 Amazon S3 存储桶。

使用 AWS CloudFormation 控制台创建对象 Lambda 接入点

1. 在 S3 对象 Lambda 默认配置中下载 AWS CloudFormation 模板 `s3objectlambda_defaultconfig.yaml`。
2. 登录 AWS 管理控制台并通过以下网址打开 AWS CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
3. 请执行以下操作之一：
   - 如果您以前从未用过 AWS CloudFormation，请在 AWS CloudFormation 主页上选择 Create stack（创建堆栈）。
   - 如果之前用过 AWS CloudFormation，请在左侧导航窗格中选择 Stacks（堆栈）。选择 Create stack（创建堆栈），然后选择 With new resources (standard) [使用新资源（标准）]。
4. 对于 Prerequisite - Prepare template（先决条件 - 准备模板），请选择 Template is ready（模板已就绪）。
5. 对于 Specify template（指定模板），选择 Upload a template file（上传模板文件）并上传 `s3objectlambda_defaultconfig.yaml`。
6. 请选择 Next（下一步）。
7. 在 Specify stack details（指定堆栈详细信息）页面上，输入堆栈名称。
8. 在 Parameters（参数）部分中，指定在堆栈模板中定义的以下参数：
   a. 对于 CreateNewSupportingAccessPoint，请执行以下操作之一：
      - 如果对于您上传了模板的 S3 存储桶已经有支持接入点，请选择 false。
      - 如果要为该存储桶创建新的接入点，请选择 true。
   b. 对于 EnableCloudWatchMonitoring，根据您是否要启用 Amazon CloudWatch 请求指标和告警，选择 true 或 false。
   c. （可选）对于 LambdaFunctionPayload，添加要提供给 Lambda 函数作为输入的 JSON 文本。您可以为调用同一 Lambda 函数的不同对象 Lambda 接入点配置具有不同参数的负载，从而扩展 Lambda 函数的灵活性。

> **Important**
> 使用对象 Lambda 接入点时，确保负载不包含任何机密信息。

d. 对于 LambdaFunctionRuntime，输入适用于 Lambda 函数的首选运行时。可用的选项为 `nodejs14.x`、`python3.9`、`java11`。

e. 对于 LambdaFunctionS3BucketName，输入您在其中上传了部署文件包的 Amazon S3 存储桶名称。

f. 对于 LambdaFunctionS3Key，输入您在上传部署文件包时使用的 Amazon S3 对象密钥。

g. 对于 LambdaFunctionS3ObjectVersion，输入您在其中上传了部署文件包的 Amazon S3 对象版本。

h. 对于 ObjectLambdaAccessPointName，输入对象 Lambda 接入点的名称。

i. 对于 S3BucketName，输入将与对象 Lambda 接入点相关联的 Amazon S3 存储桶名称。

j. 对于 SupportingAccessPointName，输入支持接入点的名称。

> **Note**
>
> 这是与您在上一步中选择的 Amazon S3 存储桶相关联的接入点。如果您没有任何与 Amazon S3 存储桶相关联的接入点，则可以配置模板，以通过为 CreateNewSupportingAccessPoint 选择 true 来为您创建一个此类接入点。

9. 请选择 Next（下一步）。

10. 在 配置堆栈选项 页面上，请选择 下一步。

    有关此页面上的可选设置的更多信息，请参阅《AWS CloudFormation 用户指南》中的设置 AWS CloudFormation 堆栈选项。

11. 在 Review（查看）页面中，请选择 Create stack（创建堆栈）。

使用 AWS Cloud Development Kit (AWS CDK)

有关使用 AWS CDK 配置对象 Lambda 接入点的更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) API 参考》中的 `AWS::S3ObjectLambda` 构造库。

使用 CloudFormation 模板自动进行 S3 对象 Lambda 设置

您可以使用 AWS CloudFormation 模板快速创建 Amazon S3 对象 Lambda 接入点。CloudFormation 模板会自动创建相关资源、配置 AWS Identity and Access Management（IAM）角色并设置 AWS Lambda 函数，该函数可通过对象 Lambda 接入点自动处理请求。使用 CloudFormation 模板，您可以实施最佳实践、改善安全状况并减少手动流程导致的错误。

此 GitHub 存储库包含 CloudFormation 模板和 Lambda 函数源代码。有关如何使用模板的说明，请参阅the section called “创建对象 Lambda 接入点”。

模板中提供的 Lambda 函数未运行任何转换。相反，它按原样从 S3 存储桶返回对象。您可以克隆函数并添加自己的转换代码，以便在数据返回到应用程序时修改和处理数据。有关修改函数的更多信息，请参阅the section called “修改 Lambda 函数”和the section called “编写 Lambda 函数”。

修改模板。

创建新的支持接入点

S3 对象 Lambda 使用两个接入点，一个对象 Lambda 接入点和一个标准 S3 接入点（称为支持接入点）。当您向对象 Lambda 接入点发出请求时，S3 会代表您调用 Lambda，或将请求委派给支持接入点，具体取决于 S3 对象 Lambda 配置。在部署模板时，您可以通过将以下参数作为 `aws cloudformation deploy` 命令的一部分来创建一个新的支持接入点。

```
CreateNewSupportingAccessPoint=true
```

配置函数有效负载

在部署模板时，可以通过将以下参数作为 `aws cloudformation deploy` 命令的一部分传递，配置有效负载为 Lambda 函数提供补充数据。

```
LambdaFunctionPayload="format=json"
```

启用 Amazon CloudWatch 监控

您可以在部署模板时通过将以下参数作为 `aws cloudformation deploy` 命令的一部分来启用 CloudWatch 监视。

```
EnableCloudWatchMonitoring=true
```

此参数将为 Amazon S3 请求指标启用对象 Lambda 接入点，并创建两个 CloudWatch 告警来监控客户端和服务器端错误。

> **Note**
>
> Amazon CloudWatch 的使用将产生额外费用。有关 Amazon S3 请求指标的更多信息，请参阅监控和记录接入点。

有关定价详细信息，请参阅 CloudWatch 定价。

配置预配置并发

要减少延迟，您可以通过编辑模板以在 Resources 之下包含以下各行，从而为支持对象 Lambda 接入点的 Lambda 函数配置预调配并发。

```
LambdaFunctionVersion:
      Type: AWS::Lambda::Version
      Properties:
        FunctionName: !Ref LambdaFunction
        ProvisionedConcurrencyConfig:
            ProvisionedConcurrentExecutions: Integer
```

> **Note**
>
> 配置并发将产生额外费用。有关预调配并发的更多信息，请参阅《AWS Lambda 开发人员指南》中的管理 Lambda 预调配并发。
> 有关定价的详细信息，请参阅 AWS Lambda 定价。

修改 Lambda 函数

更改 **GetObject** 请求的标头值

默认情况下，Lambda 函数会将所有标头（Content-Length 和 ETag 除外）从预签名 URL 请求转发到 GetObject 客户端。根据 Lambda 函数中的转换代码，您可以选择向 GetObject 客户端发送新的标头值。

您可以更新 Lambda 函数，以通过在 WriteGetObjectResponse API 操作中传递来发送新的标头值。

例如，如果 Lambda 函数将 Amazon S3 对象中的文本转换为另一种语言，则可以在 Content-Language 标头中传递一个新值。您可以通过修改 writeResponse 函数来实现，如下所示：

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);
```

```
  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    ContentLanguage: 'my-new-language'
  }).promise();
}
```

有关受支持标头的完整列表，请参阅 Amazon Simple Storage Service API 参考中的 WriteGetObjectResponse。

返回元数据标头

您可以更新 Lambda 函数，以通过在 WriteGetObjectResponse API 操作请求中传递来发送新的标头值。

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest,
      'my-new-header': 'my-new-value'
    },
    ...headers
  }).promise();
}
```

返回新状态码

您可以通过在 WriteGetObjectResponse API 操作请求中进行传递来将自定义状态代码返回至 GetObject 客户端。

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    StatusCode: Integer
  }).promise();
}
```

有关受支持的状态代码的完整列表，请参阅 Amazon Simple Storage Service API 参考中的 WriteGetObjectResponse。

将 **Range** 和 **partNumber** 参数应用于源对象

默认情况下，由 CloudFormation 模板创建的对象 Lambda 接入点可以处理 Range 和 partNumber 参数。Lambda 函数将请求的范围或段编号应用于转换后的对象。为此，此函数必须下载整个对象并运行转换。在某些情况下，转换后的对象范围可能会精确映射到源对象范围。这意味着，在源对象上请求字节范围 A-B 并运行转换，可能会产生与请求整个对象、运行转换以及在转换后的对象上返回字节范围 A-B 相同的结果。

在这种情况下，您可以更改 Lambda 函数实现，以便将范围或段编号直接应用于源对象。这种方法减少了所需的总体函数延迟和内存。有关更多信息，请参阅 the section called “使用 Range 和 partNumber 标头”。

禁用 **Range** 和 **partNumber** 处理

默认情况下，由 CloudFormation 模板创建的对象 Lambda 接入点可以处理 Range 和 partNumber 参数。如果不需要此行为，可以通过从模板中移除以下各行来禁用它。

```
AllowedFeatures:
  - GetObject-Range
  - GetObject-PartNumber
```

```
  - HeadObject-Range
  - HeadObject-PartNumber
```

转换大型对象

默认情况下，Lambda 函数会处理内存中的整个对象，然后才能开始将响应流式传输到 S3 对象 Lambda。您可以在执行转换时修改函数以流式传输响应。这样做有助于减少转换延迟和 Lambda 函数内存大小。有关实施示例，请参阅 Stream compressed content example（流式压缩内容示例）。

使用 Amazon S3 对象 Lambda 接入点

通过 Amazon S3 对象 Lambda 接入点发出请求与通过其他接入点发出请求的工作方式相同。有关如何通过接入点发出请求的更多信息，请参阅 使用接入点。您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API，通过对象 Lambda 接入点发出请求。

> **Important**
>
> 对象 Lambda 接入点的 Amazon 资源名称（ARN）使用 `s3-object-lambda` 的服务名称。因此，对象 Lambda 接入点 ARN 以 `arn:aws:s3-object-lambda`（而不是 `arn:aws:s3`）开头，它与其他接入点结合使用。

如何查找对象 Lambda 接入点的 ARN

要将对象 Lambda 接入点与 AWS CLI 或 AWS SDK 结合使用，您需要知道对象 Lambda 接入点的 Amazon 资源名称（ARN）。以下示例说明如何通过使用 Amazon S3 控制台或 AWS CLI 查找对象 Lambda 接入点的 ARN。

使用 S3 控制台

使用控制台查找对象 Lambda 接入点的 ARN

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择对象 Lambda 接入点。
3. 选择要复制其 ARN 的对象 Lambda 接入点旁边的选项按钮。
4. 请选择复制 ARN。

使用 AWS CLI

使用 AWS CLI 查找对象 Lambda 接入点的 ARN

1. 要检索与 AWS 账户 关联的对象 Lambda 接入点的列表，请运行以下命令。在运行命令之前，请将账户 ID *111122223333* 替换为您的 AWS 账户 ID。

```
aws s3control list-access-points-for-object-lambda --account-id 111122223333
```

2. 查看命令输出以查找您要使用的对象 Lambda 接入点 ARN。前一个命令的输出内容应类似如下示例。

```
{
    "ObjectLambdaAccessPointList": [
        {
            "Name": "my-object-lambda-ap",
            "ObjectLambdaAccessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/my-object-lambda-ap"
        },
        ...
    ]
}
```

如何为您的 S3 存储桶对象 Lambda 接入点使用存储桶式别名

当您创建对象 Lambda 接入点时，Amazon S3 会自动为您的对象 Lambda 接入点生成一个唯一的别名。您可以在接入点数据面板操作请求中使用此别名，而不使用 Amazon S3 存储桶名称或对象 Lambda 接入点 Amazon 资源名称（ARN）。有关这些操作的列表，请参阅接入点与AWS服务的兼容性。

对象 Lambda 接入点别名是在与 Amazon S3 存储桶相同的命名空间中创建的。此别名自动生成，无法更改。对于现有对象 Lambda 接入点，会自动分配别名以供使用。对象 Lambda 接入点别名符合有效 Amazon S3 存储桶名称的所有要求，包括以下部分：

Object Lambda Access Point name prefix-*metadata*--ol-s3

> **Note**
>
> `--ol-s3` 后缀是为对象 Lambda 接入点别名预留的，不能用于存储桶或对象 Lambda 接入点名称。有关 Amazon S3 存储桶命名规则的更多信息，请参阅存储桶命名规则。

以下示例显示了名为 *my-object-lambda-access-point* 的对象 Lambda 接入点的 ARN 和对象 Lambda 接入点别名。

- ARN – `arn:aws:s3-object-lambda:`*region*`:`*account-id*`:accesspoint/`*my-object-lambda-access-point*
- 对象 Lambda 接入点别名 – *my-object-lambda-acc*-*1a4n8yjrb3kda96f67zwrwiiuse1a*`--ol-s3`

当您使用对象 Lambda 接入点时，无需进行大量代码更改，即可使用对象 Lambda 接入点别名。

当您删除对象 Lambda 接入点时，对象 Lambda 接入点别名将变为非活动状态且未调配。

如何查找对象 Lambda 接入点的别名

使用 S3 控制台

使用控制台查找对象 Lambda 接入点的别名

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择对象 Lambda 接入点。
3. 对于要使用的对象 Lambda 接入点，复制对象 Lambda 接入点别名值。

使用 AWS CLI

创建对象 Lambda 接入点时，Amazon S3 会自动生成对象 Lambda 接入点别名，如以下示例命令所示。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。有关如何使用 AWS CLI 创建对象 Lambda 接入点的信息，请参阅使用 AWS CLI 创建对象 Lambda 接入点。

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-access-point --configuration file://my-olap-configuration.json
{
```

```
    "ObjectLambdaAccessPointArn": "arn:aws:s3:region:111122223333:accesspoint/my-
access-point",
    "Alias": {
        "Value": "my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a--ol-s3",
        "Status": "READY"
    }
}
```

生成的对象 Lambda 接入点别名有两个字段：

- `Value` 字段是对象 Lambda 接入点的别名值。
- `Status` 字段是对象 Lambda 接入点别名的状态。如果状态为 `PROVISIONING`，则 Amazon S3 正在预调配对象 Lambda 接入点别名，并且该别名尚未准备就绪，无法使用。如果状态为 `READY`，则对象 Lambda 接入点别名已成功调配，并准备就绪可供使用。

有关 REST API 中 `ObjectLambdaAccessPointAlias` 数据类型的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CreateAccessPointForObjectLambda 和 ObjectLambdaAccessPointAlias。

如何使用对象 Lambda 接入点别名

您可以使用对象 Lambda 接入点别名，而不是接入点与AWS服务的兼容性中所列的操作的 Amazon S3 存储桶名称。

以下 `get-bucket-location` 命令的 AWS CLI 示例使用存储桶的接入点别名来返回存储桶所在的 AWS 区域。要运行此命令，请将 *`user input placeholders`* 替换为您自己的信息。

```
aws s3api get-bucket-location --bucket my-object-lambda-
acc-w7i37nq6xuzgax3jw3oqtifiusw2a--ol-s3

{
    "LocationConstraint": "us-west-2"
}
```

如果请求中的对象 Lambda 接入点别名无效，则返回错误代码 `InvalidAccessPointAliasError`。有关 `InvalidAccessPointAliasError` 的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的错误代码列表。

对象 Lambda 接入点别名的限制与接入点别名的限制相同。有关接入点别名限制的更多信息，请参阅限制。

S3 对象 Lambda 接入点的安全注意事项

使用 Amazon S3 对象 Lambda，您可以借助使用 AWS Lambda 作为计算平台的规模和灵活性，在数据离开 Amazon 3 时对其执行自定义转换。S3 和 Lambda 在默认情况下保持安全，但是为了维持这种安全性，Lambda 函数的作者需要特别考虑一些注意事项。S3 对象 Lambda 要求所有访问都由经过身份验证的主体（无匿名访问）和 HTTPS 进行。

为降低安全风险，我们提出以下建议：

- 将 Lambda 执行角色的范围限定为尽可能小的权限集。
- 只要有可能，就确保您的 Lambda 函数通过提供的预签名 URL 访问 Amazon S3。

配置 IAM 策略

S3 接入点支持 AWS Identity and Access Management（IAM）资源策略，这些策略允许您按资源、用户或其他条件控制接入点的使用。有关更多信息，请参阅 为对象 Lambda 接入点配置 IAM 策略。

加密行为

由于对象 Lambda 接入点同时使用 Amazon S3 和 AWS Lambda，因此，加密行为会存在差异。有关默认 S3 加密行为的更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

- 将 S3 服务器端加密与对象 Lambda 接入点结合使用时，需要先将对象解密，然后再将其发送到 Lambda。将对象发送到 Lambda 后，将以未加密方式对其进行处理（如果是 GET 或 HEAD 请求）。
- 为防止密钥被记录，S3 将拒绝针对以下对象的 GET 和 HEAD 请求：使用具有客户提供的密钥的服务器端加密（SSE-C）来加密的对象。但是，如果 Lambda 函数可以访问客户端提供的密钥，它仍可检索这些对象。
- 将 S3 客户端加密与对象 Lambda 接入点结合使用时，请确保 Lambda 有权访问加密密钥，以便它可以解密和重新加密对象。

接入点安全性

S3 对象 Lambda 使用两个接入点，一个对象 Lambda 接入点和一个标准 S3 接入点（称为支持接入点）。当您向对象 Lambda 接入点发出请求时，S3 会代表您调用 Lambda，或将请求委派给支持接入点，具体取决于 S3 对象 Lambda 配置。当针对请求调用 Lambda 时，S3 将通过支持接入点，代表您向对象生成一个预签名 URL。您的 Lambda 函数将在调用此函数时接收此 URL 以作为输入。

您可以将 Lambda 函数设置为使用此预签名 URL 来检索原始对象，而不是直接调用 S3。通过使用此模型，您可以为对象应用更好的安全边界。您可以将通过 S3 存储桶或 S3 接入点进行的直接对象访问，限制为一组有限的 IAM 角色或用户。这种方法也可以防止您的 Lambda 函数受到混淆代理问题的影响，发生混淆代理问题时，配置错误的函数具有与调用方不同的权限，可以允许或拒绝对于对象的访问（但原本不应如此）。

对象 Lambda 接入点公共访问权限

S3 对象 Lambda 不允许匿名访问和公有访问，因为 Amazon S3 必须向您的身份授权，才能完成任何 S3 对象 Lambda 请求。当通过对象 Lambda 接入点调用请求时，您必须对于已配置的 Lambda 函数拥有 `lambda:InvokeFunction` 权限。同样，当通过对象 Lambda 接入点调用其他 API 操作时，您必须拥有所需的 `s3:*` 权限。

如果不具有这些权限，则调用 Lambda 或委托给 S3 的请求将失败，并出现 HTTP 403 (Forbidden) [HTTP 403（禁止访问）] 错误。必须由经过身份验证的主体进行所有访问。如果您需要公有访问权限，可使用 Lambda@Edge 作为可能的替代方案。有关更多信息，请参阅 Amazon CloudFront 开发人员指南中的使用 Lambda@Edge 在边缘进行自定义。

对象 Lambda 接入点 IP 地址

`describe-managed-prefix-lists` 子网支持网关虚拟私有云（VPC）端点，并与 VPC 端点的路由表相关。由于对象 Lambda 接入点不支持网关 VPC，因此其 IP 范围缺失。缺失的范围属于 Amazon S3，但网关 VPC 端点不支持。有关 `describe-managed-prefix-lists` 的更多信息，请参阅《Amazon EC2 API 参考》中的 DescribeManagedPrefixLists 和《AWS 一般参考》中的 AWS IP 地址范围。

为对象 Lambda 接入点配置 IAM 策略

Amazon S3 接入点支持 AWS Identity and Access Management（IAM）资源策略，您可以使用这些策略按资源、用户或其他条件控制接入点的使用。您可以通过对象 Lambda 接入点上的可选资源策略或支持接入点上的资源策略来控制访问权限。有关分步示例，请参阅 教程：使用 S3 对象 Lambda 转换应用程序的数据 和 教程：使用 S3 对象 Lambda 和 Amazon Comprehend 检测和修订 PII 数据。

以下四项资源必须已获得使用对象 Lambda 接入点的权限：

- IAM 身份，例如用户或角色。有关 IAM 身份和最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 身份（用户、用户组和角色）。
- 存储桶及其关联的标准接入点。当您使用对象 Lambda 接入点时，此标准接入点称为支持接入点。
- 对象 Lambda 接入点。

- AWS Lambda 函数。

> **Important**
>
> 在保存策略之前，确保解决来自 AWS Identity and Access Management Access Analyzer 的安全警告、错误、一般警告和建议。IAM Access Analyzer 将根据 IAM 策略语法和最佳实践运行策略检查，以验证您的策略。这些检查项生成结果并提供可操作的建议，可帮助您编写可操作且符合安全最佳实践的策略。
>
> 要了解有关使用 IAM Access Analyzer 验证策略的更多信息，请参阅《IAM 用户指南》中的 IAM Access Analyzer 策略验证。要查看 IAM Access Analyzer 返回的警告、错误和建议的列表，请参阅 IAM Access Analyzer 策略检查引用。

以下策略示例假设您具有以下资源：

- 具有以下 Amazon 资源名称（ARN）的 Amazon S3 存储桶：

  `arn:aws:s3:::amzn-s3-demo-bucket1`

- 此存储桶上具有以下 ARN 的 Amazon S3 standard 接入点：

  `arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point`

- 具有以下 ARN 的对象 Lambda 接入点：

  `arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-object-lambda-ap`

- 具有以下 ARN 的 AWS Lambda Lambda 函数：

  `arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction`

> **Note**
>
> 如果使用您的账户中的 Lambda 函数，则必须在策略声明中包含具体的函数版本。在以下示例 ARN 中，版本由 *1* 表示：
> `arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction:1`

Lambda 不支持向版本 $LATEST 中添加 IAM 策略。有关 Lambda 函数版本的更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 函数版本。

Example – 将访问控制委派给标准接入点的存储桶策略

下面的 S3 存储桶策略示例将存储桶的访问控制委派给存储桶的标准接入点。此策略允许对存储桶拥有者账户所拥有的所有接入点进行完全访问。因此，对此存储桶的所有访问都由附加到其接入点的策略控制。用户只能通过接入点从存储桶读取，这意味着您只能通过接入点调用操作。有关更多信息，请参阅将访问控制委派到接入点。

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS":"account-ARN"},
        "Action" : "*",
        "Resource" : [
            "arn:aws:s3:::amzn-s3-demo-bucket1",
            "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        ],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account
 ID" }
        }
    }]
}
```

Example – 向用户授予使用对象 Lambda 接入点的必需权限的 IAM 策略

以下 IAM 策略授予用户对 Lambda 函数、标准接入点和对象 Lambda 接入点的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowLambdaInvocation",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Effect": "Allow",
```

```
      "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:MyObjectLambdaFunction:1",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowStandardAccessPointAccess",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowObjectLambdaAccess",
      "Action": [
        "s3-object-lambda:Get*",
        "s3-object-lambda:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-
object-lambda-ap"
    }
  ]
}
```

启用 Lambda 执行角色的权限

当向对象 Lambda 接入点发出 GET 请求时，Lambda 函数需要将数据发送到 S3 对象 Lambda 接入点的权限。可通过对 Lambda 函数的执行角色启用 `s3-object-lambda:WriteGetObjectResponse` 权限来提供此权限。您可以创建新角色，或更新现有角色。

> **Note**
>
> 仅当您发出 GET 请求时，函数才需要 `s3-object-lambda:WriteGetObjectResponse` 权限。

在 IAM 控制台中创建执行角色

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在左侧导航窗格中，选择 Roles（角色）。
3. 选择 Create role（创建角色）。
4. 在 Common use cases（常用使用案例）下，请选择 Lambda。
5. 选择下一步。
6. 在 Add permissions（添加权限）页面上，搜索 AWS 托管式策略 `AmazonS3ObjectLambdaExecutionRolePolicy`，然后选中策略名称旁边的复选框。

   该策略应包含 `s3-object-lambda:WriteGetObjectResponse` 操作。
7. 选择下一步。
8. 在 Name, review, and create（名称、查看和创建）页面上，对于 Role name（角色名称），输入 **`s3-object-lambda-role`**。
9. （可选）为该角色添加描述和标签。
10. 选择 Create role（创建角色）。
11. 应用新创建的 **`s3-object-lambda-role`** 作为 Lambda 函数的执行角色。这可以在 Lambda 控制台中创建 Lambda 函数期间或之后完成。

有关执行角色的更多信息，请参阅 AWS Lambda 开发人员指南中的 Lambda 执行角色。

将上下文键与对象 Lambda 接入点结合使用

S3 对象 Lambda 将评估与请求的连接或签名相关的上下文键，如 `s3-object-lambda:TlsVersion` 或 `s3-object-lambda:AuthType`。Amazon S3 评估所有其他上下文键，例如 `s3:prefix`。

对象 Lambda 接入点 CORS 支持

当 S3 对象 Lambda 收到来自浏览器的请求或包含 `Origin` 标头的请求时，S3 对象 Lambda 始终会添加 `“AllowedOrigins":"*"` 标头字段。

有关更多信息，请参阅 使用跨源资源共享 (CORS)。

为 S3 对象 Lambda 接入点编写 Lambda 函数

本节详细介绍如何编写与 Amazon S3 对象 Lambda 接入点结合使用的 AWS Lambda 函数。

要了解一些 S3 对象 Lambda 任务的完整端到端过程，请参阅以下内容：

- 教程：使用 S3 对象 Lambda 转换应用程序的数据
- 教程：使用 S3 对象 Lambda 和 Amazon Comprehend 检测和修订 PII 数据
- 教程：使用 S3 对象 Lambda 在检索图像时对其动态加水印

主题

- 在 Lambda 中处理 GetObject 请求
- 在 Lambda 中处理 HeadObject 请求
- 在 Lambda 中处理 ListObjects 请求
- 在 Lambda 中处理 ListObjectsV2 请求
- 事件上下文格式和用法
- 使用 Range 和 partNumber 标头

在 Lambda 中处理 **`GetObject`** 请求

本节假设您的对象 Lambda 接入点配置为对于 `GetObject` 调用 Lambda 函数。S3 对象 Lambda 包括 Amazon S3 API 操作 `WriteGetObjectResponse`，该操作可以让 Lambda 函数向 `GetObject` 调用方提供自定义数据和响应标头。

WriteGetObjectResponse 可根据您的处理需要，让您对状态代码、响应标头和响应正文进行广泛的控制。您可以使用 WriteGetObjectResponse 来响应整个变换对象、变换对象的某些部分或其他基于应用程序上下文的响应。以下部分介绍了使用 WriteGetObjectResponse API 操作的独特示例。

- 示例 1：使用 HTTP 状态代码 403（禁止访问）进行响应
- 示例 2： 通过转换后的图像进行响应
- 示例 3： 流式压缩内容

示例 1：使用 HTTP 状态代码 403（禁止访问）进行响应

您可以使用 WriteGetObjectResponse 根据对象的内容使用 HTTP 状态代码 403（禁止）进行响应。

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import java.io.ByteArrayInputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example1 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();

        // Check to see if the request contains all of the necessary information.
        // If it does not, send a 4XX response and a custom error code and message.
        // Otherwise, retrieve the object from S3 and stream it
        // to the client unchanged.
```

```
        var tokenIsNotPresent = !
event.getUserRequest().getHeaders().containsKey("requiredToken");
        if (tokenIsNotPresent) {
            s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                    .withRequestRoute(event.outputRoute())
                    .withRequestToken(event.outputToken())
                    .withStatusCode(403)
                    .withContentLength(0L).withInputStream(new
 ByteArrayInputStream(new byte[0]))
                    .withErrorCode("MissingRequiredToken")
                    .withErrorMessage("The required token was not present in the
 request."));
            return;
        }

        // Prepare the presigned URL for use and make the request to S3.
        HttpClient httpClient = HttpClient.newBuilder().build();
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Stream the original bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(presignedResponse.body()));
    }
}
```

Python

```
import boto3
import requests

def handler(event, context):
    s3 = boto3.client('s3')

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
```

```
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    user_request_headers = event["userRequest"]["headers"]

    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    is_token_present = "SuperSecretToken" in user_request_headers

    if is_token_present:
        # If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        response = requests.get(s3_url)
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 Body=response.content)
    else:
        # If the token is not present, we send an error back to the user.
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 StatusCode=403,
        ErrorCode="NoSuperSecretTokenFound", ErrorMessage="The request was not
 secret enough.")

    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
```

```
    // The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    const { userRequest, getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    const isTokenPresent = Object
        .keys(userRequest.headers)
        .includes("SuperSecretToken");

    if (!isTokenPresent) {
        // If the token is not present, we send an error back to the user. The
 'await' in front of the request
        // indicates that we want to wait for this request to finish sending before
 moving on.
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            StatusCode: 403,
            ErrorCode: "NoSuperSecretTokenFound",
            ErrorMessage: "The request was not secret enough.",
        }).promise();
    } else {
        // If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        // Again, note the presence of 'await'.
        const presignedResponse = await axios.get(inputS3Url);
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: presignedResponse.data,
        }).promise();
    }

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

示例 2： 通过转换后的图像进行响应

执行图像转换时，您可能会发现需要源对象的所有字节，然后才能开始处理它们。在这种情况下，您的 WriteGetObjectResponse 请求会在一次调用中将整个对象返回给请求的应用程序。

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.awt.Image;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example2 {

    private static final int HEIGHT = 250;
    private static final int WIDTH = 250;

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Prepare the presigned URL for use and make the request to S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // The entire image is loaded into memory here so that we can resize it.
        // Once the resizing is completed, we write the bytes into the body
        // of the WriteGetObjectResponse request.
        var originalImage = ImageIO.read(presignedResponse.body());
```

```
        var resizingImage = originalImage.getScaledInstance(WIDTH, HEIGHT,
 Image.SCALE_DEFAULT);
        var resizedImage = new BufferedImage(WIDTH, HEIGHT,
 BufferedImage.TYPE_INT_RGB);
        resizedImage.createGraphics().drawImage(resizingImage, 0, 0, WIDTH, HEIGHT,
 null);

        var baos = new ByteArrayOutputStream();
        ImageIO.write(resizedImage, "png", baos);

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(new ByteArrayInputStream(baos.toByteArray())));
    }
}
```

Python

```
import boto3
import requests
import io
from PIL import Image

def handler(event, context):
    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    """
    In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
```

```
    'inputS3Url'.
    """
    image_request = requests.get(s3_url)
    image = Image.open(io.BytesIO(image_request.content))
    image.thumbnail((256,256), Image.ANTIALIAS)

    transformed = io.BytesIO()
    image.save(transformed, "png")

    # Send the resized image back to the client.
    s3 = boto3.client('s3')
    s3.write_get_object_response(Body=transformed.getvalue(), RequestRoute=route,
 RequestToken=token)

    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const sharp = require('sharp');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    // 'inputS3Url'.
    const { data } = await axios.get(inputS3Url, { responseType: 'arraybuffer' });

    // Resize the image.
    const resized = await sharp(data)
        .resize({ width: 256, height: 256 })
        .toBuffer();
```

```
    // Send the resized image back to the client.
    await s3.writeGetObjectResponse({
        RequestRoute: outputRoute,
        RequestToken: outputToken,
        Body: resized,
    }).promise();

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

示例 3： 流式压缩内容

压缩对象时，压缩数据是以增量方式生成的。因此，您可以使用您的 `WriteGetObjectResponse` 请求在压缩数据准备就绪后立即返回压缩的数据。如本示例所示，您不需要知道已完成转换的长度。

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.WriteGetObjectResponseRequest;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example3 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        AmazonS3 s3Client = AmazonS3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Request the original object from S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());
```

```
        // Consume the incoming response body from the presigned request,
        // apply our transformation on that data, and emit the transformed bytes
        // into the body of the WriteGetObjectResponse request as soon as they're
 ready.
        // This example compresses the data from S3, but any processing pertinent
        // to your application can be performed here.
        var bodyStream = new GZIPCompressingInputStream(presignedResponse.body());

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(new WriteGetObjectResponseRequest()
                .withRequestRoute(event.outputRoute())
                .withRequestToken(event.outputToken())
                .withInputStream(bodyStream));
    }

}
```

Python

```
import boto3
import requests
import zlib
from botocore.config import Config


"""
A helper class to work with content iterators. Takes an interator and compresses the
 bytes that come from it. It
implements 'read' and '__iter__' so that the SDK can stream the response.
"""
class Compress:
    def __init__(self, content_iter):
        self.content = content_iter
        self.compressed_obj = zlib.compressobj()

    def read(self, _size):
        for data in self.__iter__()
            return data

    def __iter__(self):
        while True:
            data = next(self.content)
```

```
            chunk = self.compressed_obj.compress(data)
            if not chunk:
                break

            yield chunk

        yield self.compressed_obj.flush()


def handler(event, context):
    """
    Setting the 'payload_signing_enabled' property to False allows us to send a
 streamed response back to the client.
    in this scenario, a streamed response means that the bytes are not buffered into
 memory as we're compressing them,
    but instead are sent straight to the user.
    """
    my_config = Config(
        region_name='eu-west-1',
        signature_version='s3v4',
        s3={
            "payload_signing_enabled": False
        }
    )
    s3 = boto3.client('s3', config=my_config)

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Compress the 'get' request stream.
    with requests.get(s3_url, stream=True) as r:
        compressed = Compress(r.iter_content())

        # Send the stream back to the client.
```

```
        s3.write_get_object_response(Body=compressed, RequestRoute=route,
 RequestToken=token, ContentType="text/plain",
                                     ContentEncoding="gzip")

    # Gracefully exit the Lambda function.
    return {'status_code': 200}
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const zlib = require('zlib');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // Download the object from S3 and process it as a stream, because it might be a
 huge object and we don't want to
    // buffer it in memory. Note the use of 'await' because we want to wait for
 'writeGetObjectResponse' to finish
    // before we can exit the Lambda function.
    await axios({
        method: 'GET',
        url: inputS3Url,
        responseType: 'stream',
    }).then(
        // Gzip the stream.
        response => response.data.pipe(zlib.createGzip())
    ).then(
        // Finally send the gzip-ed stream back to the client.
        stream => s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: stream,
            ContentType: "text/plain",
            ContentEncoding: "gzip",
```

```
        }).promise()
    );

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

> **Note**
>
> 虽然 S3 对象 Lambda 允许使用长达 60 秒的时间通过 `WriteGetObjectResponse` 请求来将完整响应发送给发起人，但实际可用时间可能会减少。例如，Lambda 函数超时可能少于 60 秒。在其他情况下，发起人可能会有更严格的超时。

要使原始调用方能够收到非 HTTP 状态代码 500（内部服务器错误）响应，必须完成 `WriteGetObjectResponse` 调用。如果 Lambda 函数返回时引发了异常或其他情况，则在调用 `WriteGetObjectResponse` API 操作之前，原始调用方将收到 500（内部服务器错误）响应。在完成响应所需的时间内引发的异常将导致截断对发起人的响应。如果 Lambda 函数从 `WriteGetObjectResponse` API 调用收到 HTTP 状态代码 200（OK）响应，即表示原始调用方已经发送了完整的请求。S3 对象 Lambda 会忽略 Lambda 函数的响应，无论是否抛出异常。

调用 `WriteGetObjectResponse` API 操作时，Amazon S3 需要事件上下文中的路由和请求令牌。有关更多信息，请参阅 事件上下文格式和用法。

将 `WriteGetObjectResult` 请求与原始调用方连接起来需要这些路由和请求令牌参数。尽管重试 500（内部服务器错误）响应始终是适当的，但请注意，由于请求令牌是一个单一用途令牌，后续尝试使用它可能会导致 HTTP 状态代码 400（错误请求）响应。虽然使用路由和请求令牌调用 `WriteGetObjectResponse` 不需要从被调用的 Lambda 函数发出，但它必须由同一账户中的身份发出。还必须在 Lambda 函数完成执行之前完成调用。

在 Lambda 中处理 **`HeadObject`** 请求

本节假设您的对象 Lambda 接入点配置为对于 `HeadObject` 调用 Lambda 函数。Lambda 将收到一个 JSON 有效负载，其中包含名为 `headObjectContext` 的密钥。在上下文中，有一个名为 `inputS3Url` 的属性，它是 `HeadObject` 的支持接入点的预签名 URL。

预签名 URL 将包含以下属性（如果已指定）：

- `versionId`（在查询参数中）

- requestPayer（在 x-amz-request-payer 标头中）
- expectedBucketOwner（在 x-amz-expected-bucket-owner 标头中）

其他属性未预签名，因此不会包含在内。当调用在 userRequest 标头中找到的预签名 URL 时，可以手动将作为标头发送的未签名选项添加到请求中。对于 HeadObject，不支持服务器端加密选项。

有关请求语法 URI 参数，请参阅《Amazon Simple Storage Service API 参考》中的 HeadObject。

以下示例显示了 HeadObject 的 Lambda JSON 输入有效负载。

```
{
  "xAmzRequestId": "requestId",
  "**headObjectContext**": {
    "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>"
  },
  "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
  },
  "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
   },
   "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
             "attributes": {
             "mfaAuthenticated": "false",
```

```
            "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
        },
        "sessionIssuer": {
            "type": "Role",
            "principalId": "principalId",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

您的 Lambda 函数应返回一个 JSON 对象，其中包含将针对 `HeadObject` 调用返回的标头和值。

下面的示例显示 `HeadObject` 的 Lambda 响应 JSON 的结构。

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "headers": {
        "Accept-Ranges": <string>,
        "x-amz-archive-status": <string>,
        "x-amz-server-side-encryption-bucket-key-enabled": <boolean>,
        "Cache-Control": <string>,
        "Content-Disposition": <string>,
        "Content-Encoding": <string>,
        "Content-Language": <string>,
        "Content-Length": <number>, // Required
        "Content-Type": <string>,
        "x-amz-delete-marker": <boolean>,
        "ETag": <string>,
        "Expires": <string>,
        "x-amz-expiration": <string>,
        "Last-Modified": <string>,
        "x-amz-missing-meta": <number>,
        "x-amz-object-lock-mode": <string>,
        "x-amz-object-lock-legal-hold": <string>,
        "x-amz-object-lock-retain-until-date": <string>,
        "x-amz-mp-parts-count": <number>,
        "x-amz-replication-status": <string>,
        "x-amz-request-charged": <string>,
```

```
        "x-amz-restore": <string>,
        "x-amz-server-side-encryption": <string>,
        "x-amz-server-side-encryption-customer-algorithm": <string>,
        "x-amz-server-side-encryption-aws-kms-key-id": <string>,
        "x-amz-server-side-encryption-customer-key-MD5": <string>,
        "x-amz-storage-class": <string>,
        "x-amz-tagging-count": <number>,
        "x-amz-version-id": <string>,
        <x-amz-meta-headers>: <string>, // user-defined metadata
        "x-amz-meta-meta1": <string>, // example of the user-defined metadata header,
 it will need the x-amz-meta prefix
        "x-amz-meta-meta2": <string>
        ...
    };
}
```

以下示例显示如何在返回 JSON 之前根据需要修改标头值，以使用预签名 URL 来填充您的响应。

Python

```
import requests

def lambda_handler(event, context):
    print(event)

    # Extract the presigned URL from the input.
    s3_url = event["headObjectContext"]["inputS3Url"]

    # Get the head of the object from S3.
    response = requests.head(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        return {
            "statusCode": response.status_code,
            "errorCode": "RequestFailure",
            "errorMessage": "Request to S3 failed"
    }

    # Store the headers in a dictionary.
    response_headers = dict(response.headers)

    # This obscures Content-Type in a transformation, it is optional to add
```

```
    response_headers["Content-Type"] = ""

    # Return the headers to S3 Object Lambda.
    return {
        "statusCode": response.status_code,
        "headers": response_headers
        }
```

在 Lambda 中处理 **ListObjects** 请求

本节假设您的对象 Lambda 接入点配置为对于 `ListObjects` 调用 Lambda 函数。Lambda 将收到带有名为 `listObjectsContext` 的新对象的 JSON 有效负载。`listObjectsContext` 包含单个属性 `inputS3Url`，它是 `ListObjects` 的支持接入点的预签名 URL。

与 `GetObject` 和 `HeadObject` 不同，预签名 URL 将包含以下属性（如果已指定）：

- 所有查询参数
- `requestPayer`（在 `x-amz-request-payer` 标头中）
- `expectedBucketOwner`（在 `x-amz-expected-bucket-owner` 标头中）

有关请求语法 URI 参数，请参阅《Amazon Simple Storage Service API 参考》中的 `ListObjects`。

> **Important**
>
> 我们建议您在开发应用程序时使用较新的版本 ListObjectsV2。为了向后兼容，Amazon S3 仍继续支持 `ListObjects`。

以下示例显示了 `ListObjects` 的 Lambda JSON 输入有效负载。

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsContext**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
```

```
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

您的 Lambda 函数应返回一个 JSON 对象，其中包含将从 S3 对象 Lambda 返回的状态码、列表 XML 结果或错误信息。

S3 对象 Lambda 不处理或验证 `listResultXml`，而是将其转发给 `ListObjects` 调用方。对于 `listBucketResult`，S3 对象 Lambda 期望某些属性属于特定类型，如果它无法解析它们，则会引发异常。无法同时提供 `listResultXml` 和 `listBucketResult`。

以下示例演示如何使用预签名 URL 调用 Amazon S3 并使用结果填充响应，包括错误检查。

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsContext"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
```

```
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

下面的示例显示 `ListObjects` 的 Lambda 响应 JSON 的结构。

```
{
  "statusCode": <number>; // Required
  "errorCode": <string>;
  "errorMessage": <string>;
  "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

  "listBucketResult": {  // listBucketResult can be provided instead of listResultXml,
 however they can not both be provided in the JSON response
        "name": <string>,  // Required for 'listBucketResult'
        "prefix": <string>,
        "marker": <string>,
        "nextMarker": <string>,
        "maxKeys": <int>,   // Required for 'listBucketResult'
        "delimiter": <string>,
```

```
        "encodingType": <string>
        "isTruncated": <boolean>,  // Required for 'listBucketResult'
        "contents": [  {
            "key": <string>,  // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>,   // CRC32,  CRC32C,  SHA1,  SHA256
            "size": <int>,   // Required for 'content'
            "owner": {
                "displayName": <string>,  // Required for 'owner'
                "id": <string>,  // Required for 'owner'
            },
            "storageClass": <string>
            },
        ...
        ],
        "commonPrefixes": [  {
            "prefix": <string>   // Required for 'commonPrefix'
        },
        ...
        ],
    }
}
```

在 Lambda 中处理 **ListObjectsV2** 请求

本节假设您的对象 Lambda 接入点配置为对于 `ListObjectsV2` 调用 Lambda 函数。Lambda 将收到带有名为 `listObjectsV2Context` 的新对象的 JSON 有效负载。`listObjectsV2Context` 包含单个属性 `inputS3Url`，它是 `ListObjectsV2` 的支持接入点的预签名 URL。

与 `GetObject` 和 `HeadObject` 不同，预签名 URL 将包含以下属性（如果已指定）：

- 所有查询参数
- `requestPayer`（在 `x-amz-request-payer` 标头中）
- `expectedBucketOwner`（在 `x-amz-expected-bucket-owner` 标头中）

有关请求语法 URI 参数，请参阅《Amazon Simple Storage Service API 参考》中的 `ListObjectsV2`。

以下示例显示了 `ListObjectsV2` 的 Lambda JSON 输入有效负载。

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsV2Context**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?list-type=2&X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
```

```
  "protocolVersion": "1.00"
}
```

您的 Lambda 函数应返回一个 JSON 对象，其中包含将从 S3 对象 Lambda 返回的状态码、列表 XML 结果或错误信息。

S3 对象 Lambda 不处理或验证 `listResultXml`，而是将其转发给 `ListObjectsV2` 调用方。对于 `listBucketResult`，S3 对象 Lambda 期望某些属性属于特定类型，如果它无法解析它们，则会引发异常。无法同时提供 `listResultXml` 和 `listBucketResult`。

以下示例演示如何使用预签名 URL 调用 Amazon S3 并使用结果填充响应，包括错误检查。

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsV2Context"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)
```

```
    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

下面的示例显示 `ListObjectsV2` 的 Lambda 响应 JSON 的结构。

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
```

```
    "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

    "listBucketResult": {  // listBucketResult can be provided instead of
 listResultXml, however they can not both be provided in the JSON response
        "name": <string>, // Required for 'listBucketResult'
        "prefix": <string>,
        "startAfter": <string>,
        "continuationToken": <string>,
        "nextContinuationToken": <string>,
        "keyCount": <int>, // Required for 'listBucketResult'
        "maxKeys": <int>, // Required for 'listBucketResult'
        "delimiter": <string>,
        "encodingType": <string>
        "isTruncated": <boolean>, // Required for 'listBucketResult'
        "contents": [ {
            "key": <string>, // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>, // CRC32, CRC32C, SHA1, SHA256
            "size": <int>, // Required for 'content'
            "owner": {
                "displayName": <string>, // Required for 'owner'
                "id": <string>, // Required for 'owner'
            },
            "storageClass": <string>
            },
            ...
        ],
        "commonPrefixes": [ {
            "prefix": <string> // Required for 'commonPrefix'
            },
        ...
        ],
    }
}
```

事件上下文格式和用法

Amazon S3 对象 Lambda 提供了有关在传递给 AWS Lambda 函数的事件中发出的请求的上下文。以下屏幕截图显示一个示例请求。示例后面包含对各个字段的描述。

```
{
```

```
    "xAmzRequestId": "requestId",
    "getObjectContext": {
        "inputS3Url": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>",
        "outputRoute": "io-use1-001",
        "outputToken": "OutputToken"
    },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
```

```
    "protocolVersion": "1.00"
}
```

请求中包含以下字段：

- `xAmzRequestId` – 此请求的 Amazon S3 请求 ID。我们建议您记录此值以帮助调试。
- `getObjectContext` – 连接到 Amazon S3 和 S3 对象 Lambda 的输入和输出详细信息。
  - `inputS3Url` – 可用于从 Amazon S3 获取原始对象的预签名 URL。URL 是使用原始调用方的身份进行签名的，使用 URL 时该用户的权限将适用。如果 URL 中有签名标头，则 Lambda 函数必须将这些标头包含在对 Amazon S3 的调用中，`Host` 标头除外。
  - `outputRoute` – 在 Lambda 函数调用 `WriteGetObjectResponse` 时添加到 S3 对象 Lambda URL 的路由令牌。
  - `outputToken` – S3 对象 Lambda 使用的不透明令牌，用于将 `WriteGetObjectResponse` 调用与原始调用方相匹配。
- `configuration` – 有关对象 Lambda 接入点的配置信息。
  - `accessPointArn` – 收到此请求的对象 Lambda 接入点的 Amazon 资源名称（ARN）。
  - `supportingAccessPointArn` – 在对象 Lambda 接入点配置中指定的支持接入点的 ARN。
  - `payload` – 应用于对象 Lambda 接入点配置的自定义数据。S3 对象 Lambda 将此数据视为不透明字符串，因此在使用前可能需要对其进行解码。
- `userRequest` – 有关对 S3 对象 Lambda 的原始调用的信息。
  - `url` – S3 对象 Lambda 接收的请求的解码 URL，不包括任何与授权相关的查询参数。
  - `headers` – 字符串到字符串的映射，包含原始调用中的 HTTP 标头及其值，不包括任何与授权相关的标头。如果同一个标头多次出现，会将相同标头的每个实例中的值组合成一个以逗号分隔的列表。此映射中会保留原始标头的大小写。
- `userIdentity` – 有关对 S3 对象 Lambda 发出调用的身份的详细信息。有关更多信息，请参阅 AWS CloudTrail 用户指南中的记录数据事件以便跟踪。
  - `type` – 身份的类型。
  - `accountId` – 身份所属的 AWS 账户。
  - `userName` – 已发出调用的身份的友好名称。
  - `principalId` – 已发出调用的身份的唯一标识符。
  - `arn` – 已发出调用的主体的 ARN。ARN 的最后一个部分包含已发出调用的用户或角色。
  - `sessionContext` – 如果已使用临时安全证书发出请求，此元素提供有关已为这些证书创建的会话的信息。

  - `invokedBy` – 发出请求的 AWS 服务 的名称，例如 Amazon EC2 Auto Scaling 或 AWS Elastic Beanstalk。
  - `sessionIssuer` – 如果已使用临时安全证书发出请求，此元素提供有关证书获取方式的信息。
- `protocolVersion` – 提供的上下文的版本 ID。此字段的格式为 `{Major Version}.{Minor Version}`。次要版本号始终是两位数字。对字段的语义进行任何删除或更改都将导致主要版本冲突，并且需要主动选择加入。Amazon S3 可以随时添加新字段，此时您可能会遇到次要版本冲突。由于软件部署的性质，您可能会同时看到多个次要版本处于使用状态。

使用 Range 和 partNumber 标头

在 Amazon S3 对象 Lambda 中使用大型对象时，您可以使用 `Range` HTTP 标头从对象中下载指定的字节范围。要从相同对象中提取不同的字节范围，您可以使用到 Amazon S3 的并发连接。还可以指定 `partNumber` 参数（1 到 10000 之间的整数），它对于对象的指定分段执行范围内的请求。

由于您可能希望通过多种方法来处理包括 `Range` 或 `partNumber` 参数的请求，S3 对象 Lambda 不会将这些参数应用于转换后的对象。相反，您的 AWS Lambda 函数必须根据应用程序的需要实施此功能。

要对 S3 对象 Lambda 使用 `Range` 和 `partNumber` 参数，请执行以下操作：

- 在对象 Lambda 接入点配置中启用这些参数。
- 编写一个 Lambda 函数，以处理包含这些参数的请求。

以下步骤介绍了如何完成此操作。

步骤 1：配置对象 Lambda 接入点

默认情况下，对象 Lambda 接入点会对在标头或查询参数中包含 `Range` 或 `partNumber` 参数的任何 `GetObject` 或 `HeadObject` 请求响应 HTTP 状态代码 501（未实施）错误。

要使对象 Lambda 接入点能够接受此类请求，您必须在对象 Lambda 接入点配置的 `AllowedFeatures` 部分中加入 `GetObject-Range`、`GetObject-PartNumber`、`HeadObject-Range` 或 `HeadObject-PartNumber`。有关更新对象 Lambda 接入点配置的更多信息，请参阅创建对象 Lambda 接入点。

步骤 2：在 Lambda 函数中实施 **Range** 或 **partNumber** 处理

当对象 Lambda 接入点使用范围内的 GetObject 或 HeadObject 请求调用 Lambda 函数时，Range 或 partNumber 参数将包含在事件上下文中。如下表所述，参数在事件上下文中的位置取决于使用的参数以及如何将其包含在对于对象 Lambda 接入点的原始请求中。

参数	事件上下文位置
Range（标头）	userRequest.headers.Range
Range（查询参数）	userRequest.url （查询参数 Range）
partNumber	userRequest.url （查询参数 partNumber ）

> **Important**
>
> 为对象 Lambda 接入点提供的预签名 URL 不包含原始请求中的 Range 或 partNumber 参数。请参阅以下选项，了解如何在 AWS Lambda 函数中处理这些参数。

在您提取 Range 或 partNumber 值后，您可以根据应用程序的需要采取以下方法之一：

A. 将请求的 **Range** 或 **partNumber** 映射到转换后的对象（建议）。

处理 Range 或 partNumber 请求的最可靠方法是执行以下操作：

- 从 Amazon S3 中检索完整的对象。
- 转换对象。
- 将请求的 Range 或 partNumber 参数应用于转换的对象。

为此，请使用提供的预签名 URL 从 Amazon S3 获取整个对象，然后根据需要处理该对象。对于 Lambda 函数以此方式处理 Range 函数的示例，请参阅 AWS Samples GitHub 存储库中的此示例。

B. 将请求的 **Range** 映射到预签名的 URL。

在某些情况下，您的 Lambda 函数可以将请求的 Range 直接映射到预签名 URL，以便仅从 Amazon S3 中检索对象的一部分。仅当您的转换符合以下两个条件时，此方法才适用：

1. 您的转换函数可以应用于部分对象范围。
2. 在转换函数之前或之后应用 Range 参数会得到相同的已转换对象。

例如，将 ASCII 编码对象中的所有字符转换为大写的转换函数符合上述两个条件。转换可以应用于对象的一部分，在转换之前应用 Range 参数获得的结果与在转换后应用此参数相同。

相比之下，反转 ASCII 编码对象中的字符的函数不符合这些条件。这样的函数符合标准 1，因为它可以应用于部分对象范围。但是，它不符合标准 2，因为在转换之前应用 Range 参数达到的结果与在转换后应用参数不同。

考虑请求将函数应用于包含内容 abcdefg 的对象的前三个字符。在转换前应用 Range 参数将仅检索 abc，然后反转数据，从而返回 cba。但是，如果在转换之后应用该参数，则函数将检索整个对象，将其反转，然后应用 Range 参数，以返回 gfe。由于这些结果不同，此函数在从 Amazon S3 中检索对象时不应应用 Range 参数。相反，它应该检索整个对象，执行转换，然后仅应用 Range 参数。

> **Warning**
>
> 在很多情况下，向预签名 URL 应用 Range 参数将导致 Lambda 函数或发出请求的客户端出现意外行为。除非您确定应用程序在从 Amazon S3 中仅检索部分对象时能够正常工作，否则我们建议您按照前面的方法 A 中所述检索和转换完整对象。

如果您的应用程序符合前面在方法 B 中介绍的条件，您可以简化您的 AWS Lambda 函数，方法是仅获取所请求的对象范围，然后在该范围内运行转换。

以下 Java 代码示例演示如何执行以下操作：

- 从 GetObject 请求中检索 Range 标头。
- 将 Range 标头添加到预签名 URL 中，Lambda 可以使用此标头从 Amazon S3 中检索请求的范围。

```
private HttpRequest.Builder applyRangeHeader(ObjectLambdaEvent event,
 HttpRequest.Builder presignedRequest) {
    var header = event.getUserRequest().getHeaders().entrySet().stream()
            .filter(e -> e.getKey().toLowerCase(Locale.ROOT).equals("range"))
            .findFirst();

    // Add check in the query string itself.
```

```
    header.ifPresent(entry -> presignedRequest.header(entry.getKey(),
 entry.getValue()));
    return presignedRequest;
}
```

使用 AWS 构建的 Lambda 函数

AWS 提供了一些预构建的 AWS Lambda 函数，您可以将这些函数与 S3 对象 Lambda 结合使用，以检测和编辑个人身份信息（PII）以及解压缩 S3 对象。这些 Lambda 函数可在 AWS Serverless Application Repository 中使用。您可以在创建对象 Lambda 接入点时通过 AWS Management Console 选择这些函数。

有关如何从 AWS Serverless Application Repository 部署无服务器应用程序的更多信息，请参阅《AWS Serverless Application Repository 开发人员指南》中的部署应用程序。

> **Note**
>
> 以下示例只能与 `GetObject` 请求结合使用。

示例 1：PII 访问控制

此 Lambda 函数使用 Amazon Comprehend，这是一项自然语言处理（NLP）服务，它使用机器学习发现文本中的见解和关系。此函数自动从 Amazon S3 存储桶的文档中检测个人身份信息（PII），例如姓名、地址、日期、信用卡号和社会保障号码。如果存储桶中具有包含 PII 的文档，则可以将 PII 访问控制函数配置为检测这些 PII 实体类型并限制未授权用户进行访问。

要开始使用，请在您的账户中部署以下 Lambda 函数，然后向对象 Lambda 接入点配置中添加此函数的 Amazon 资源名称（ARN）。

以下是此函数的示例 ARN：

```
arn:aws:serverlessrepo:us-east-1:111122223333:applications/
ComprehendPiiAccessControlS3ObjectLambda
```

您可以使用以下 AWS Serverless Application Repository 链接在 AWS Management Console上添加或查看此函数：ComprehendPiiAccessControlS3ObjectLambda。

要在 GitHub 上查看此函数，请参阅 Amazon Comprehend S3 对象 Lambda。

示例 2：PII 编辑

此 Lambda 函数使用 Amazon Comprehend，这是一项自然语言处理（NLP）服务，它使用机器学习发现文本中的见解和关系。此函数自动从 Amazon S3 存储桶的文档中编辑个人身份信息（PII），例如姓名、地址、日期、信用卡号和社会保障号码。

如果存储桶中具有包含信用卡号或银行账户信息等信息的文档，则可以将 PII Redction S3 对象 Lambda 函数配置为检测 PII，然后返回这些文档的副本，在这些文档中，已对 PII 实体类型进行了编辑。

要开始使用，请在您的账户中部署以下 Lambda 函数，然后向对象 Lambda 接入点配置中添加此函数的 ARN。

以下是此函数的示例 ARN：

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/
ComprehendPiiRedactionS3ObjectLambda
```

您可以使用以下 AWS Management Console 链接在 AWS Serverless Application Repository上添加或查看此函数：ComprehendPiiRedactionS3ObjectLambda。

要在 GitHub 上查看此函数，请参阅 Amazon Comprehend S3 对象 Lambda。

要了解 PII 编辑中某些 S3 对象 Lambda 任务的完整端到端过程，请参阅教程：使用 S3 对象 Lambda 和 Amazon Comprehend 检测和修订 PII 数据。

示例 3：Decompression

Lambda 函数 `S3ObjectLambdaDecompression` 可以将存储在 Amazon S3 中的对象解压缩为六种压缩文件格式之一：`bzip2`、`gzip`、`snappy`、`zlib`、`zstandard` 和 `ZIP`。

要开始使用，请在您的账户中部署以下 Lambda 函数，然后向对象 Lambda 接入点配置中添加此函数的 ARN。

以下是此函数的示例 ARN：

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/S3ObjectLambdaDecompression
```

您可以使用以下 AWS Management Console 链接在 AWS Serverless Application Repository上添加或查看此函数：S3ObjectLambdaDecompression。

要在 GitHub 上查看此函数，请参阅 S3 对象 Lambda 解压缩。

S3 对象 Lambda 的最佳实践和指南

使用 S3 对象 Lambda 时，请遵循以下最佳实践和指南来优化操作和性能。

主题

使用 S3 对象 Lambda

S3 对象 Lambda 仅支持处理 GET、LIST 和 HEAD 请求。任何其他请求都不会调用 AWS Lambda，而是返回标准的未转换 API 响应。每个 AWS 账户、每个区域最多可以创建 1000 个对象 Lambda 接入点。您使用的 AWS Lambda 函数必须与对象 Lambda 接入点位于同一 AWS 账户和区域中。

S3 对象 Lambda 允许使用长达 60 秒的时间将完整的响应流式传输到发起人。您的函数还受 AWS Lambda 默认限额的约束。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 配额。

当 S3 对象 Lambda 调用您指定的 Lambda 函数时，您有责任确保指定的 Lambda 函数或应用程序从 Amazon S3 中覆盖或删除的任何数据都符合预期且正确。

您只能使用 S3 对象 Lambda 对于对象执行操作。您不能使用 S3 对象 Lambda 执行其他 Amazon S3 操作，例如修改或删除存储桶。有关支持接入点的 S3 操作的完整列表，请参阅接入点与 S3 操作的兼容性。

除此列表外，对象 Lambda 接入点不支持 `POST Object`、`CopyObject`（作为源）和 `SelectObjectContent` API 操作。

与 S3 对象 Lambda 结合使用的 AWS 服务

S3 对象 Lambda 可以连接 Amazon S3 和 AWS Lambda，也可以连接您选择的其他 AWS 服务，以交付与发出请求的应用程序相关的对象。与 S3 对象 Lambda 结合使用的所有 AWS 服务将继续受其各自

服务水平协议（SLA）管控。例如，如有任何 AWS 服务不遵守其服务承诺，您将有资格获得该服务的 SLA 中介绍的服务积分。

`Range` 和 `partNumber` 标头

使用大型对象时，您可以使用 `Range` HTTP 标头从对象中下载指定的字节范围。当您使用 `Range` 标头时，您的请求仅提取对象的指定部分。您还可以使用 `partNumber` 标头对于对象中的指定分段执行范围内的请求。

有关更多信息，请参阅使用 Range 和 partNumber 标头。

转换 `expiry-date`

您可以在 AWS Management Console上从对象 Lambda 接入点打开或下载转换后的对象。这些对象必须未过期。如果您的 Lambda 函数转换了对象的 `expiry-date`，您可能会看到无法打开或下载的过期对象。此行为仅适用于 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 还原的对象。

使用 AWS CLI 和 AWS SDK

不支持将 AWS Command Line Interface（AWS CLI）S3 子命令（`cp`、`mv` 和 `sync`）和 AWS SDK for Java `TransferManager` 类与 S3 对象 Lambda 结合使用。

S3 对象 Lambda 教程

以下教程提供了一些 S3 对象 Lambda 任务的完整端到端过程。

- 教程：使用 S3 对象 Lambda 转换应用程序的数据
- 教程：使用 S3 对象 Lambda 和 Amazon Comprehend 检测和修订 PII 数据
- 教程：使用 S3 对象 Lambda 在检索图像时对其动态加水印

调试 S3 对象 Lambda

当 Lambda 函数调用或执行出现问题时，对 Amazon S3 对象 Lambda 接入点的请求可能会导致新的错误响应。这些错误的格式与标准 Amazon S3 错误的格式相同。有关 S3 对象 Lambda 错误的信息，请参阅 Amazon Simple Storage Service API 参考中的 S3 对象 Lambda 错误代码列表。

有关常规 Lambda 函数调试的更多信息，请参阅 AWS Lambda 开发人员指南中的监控和排查 Lambda 应用程序故障。

有关标准 Amazon S3 错误的信息，请参阅 Amazon Simple Storage Service API 参考中的错误响应。

您可以在 Amazon CloudWatch 中为对象 Lambda 接入点启用请求指标。这些指标可帮助您监控接入点的运行性能。您可以在创建对象 Lambda 接入点期间或之后启用请求指标。有关更多信息，请参阅 CloudWatch 中的 S3 对象 Lambda 请求指标。

要获取有关向对象 Lambda 接入点发出的请求的更精细的日志记录，您可以启用 AWS CloudTrail 数据事件。有关更多信息，请参阅 AWS CloudTrail 用户指南中的记录数据事件以便跟踪。

有关 S3 对象 Lambda 教程，请参阅以下内容：

- 教程：使用 S3 对象 Lambda 转换应用程序的数据
- 教程：使用 S3 对象 Lambda 和 Amazon Comprehend 检测和修订 PII 数据
- 教程：使用 S3 对象 Lambda 在检索图像时对其动态加水印

有关标准接入点的更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。

有关使用存储桶的信息，请参阅 存储桶概述。有关使用对象的信息，请参阅 Amazon S3 对象概述。

什么是 S3 Express One Zone？

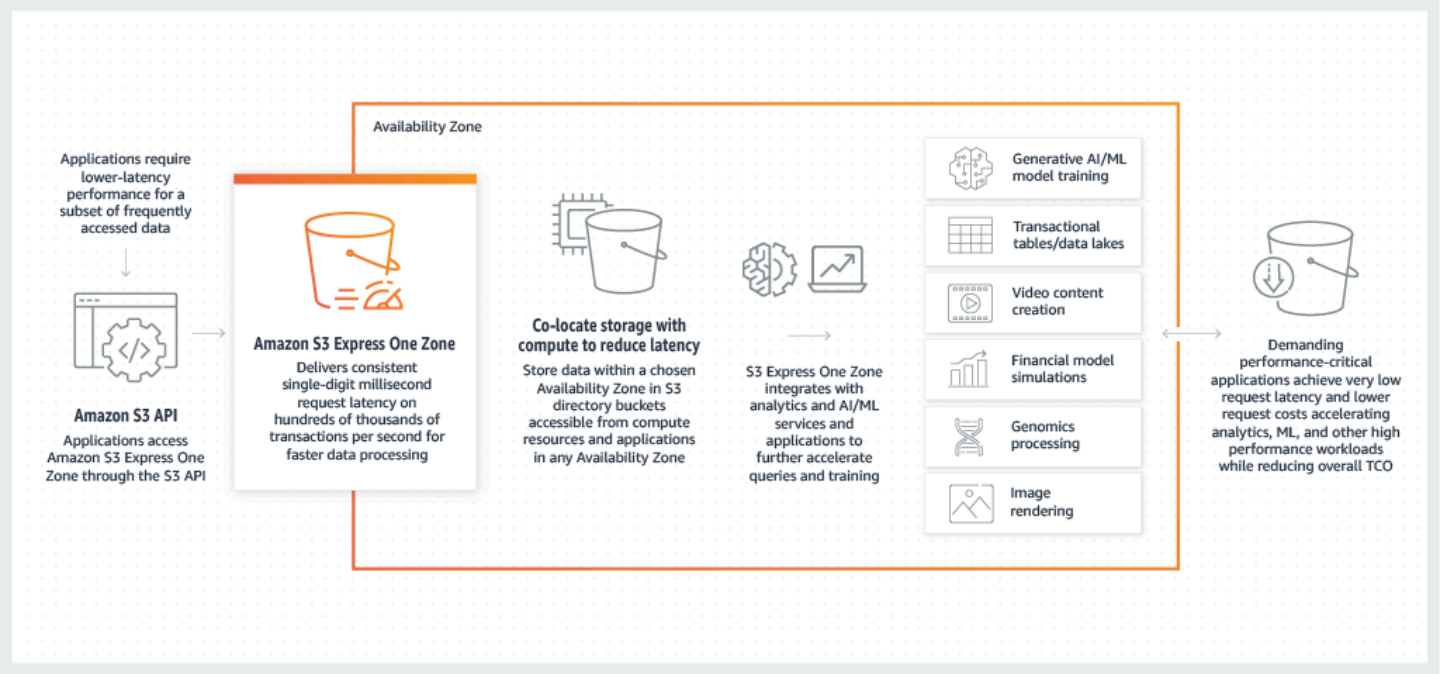
Amazon S3 Express One Zone 是高性能的单区 Amazon S3 存储类，专门用于为延迟要求极高的应用程序提供稳定的毫秒级数据访问。S3 Express One Zone 是目前具有极低延迟的云对象存储类，相比 S3 Standard，其数据访问速度要快 10 倍，且请求成本低 50%。请求的完成速度实现了数量级的提升，应用程序可以直接从中获益。S3 Express One Zone 提供与其它 S3 存储类相似的性能弹性。

与 Amazon S3 存储类一样，您无需事先规划或预调配容量或吞吐量需求。您可以根据需要纵向扩展或缩减，并通过 Amazon S3 API 访问数据。

S3 Express One Zone 是第一种可以在其中选择单个可用区的 S3 存储类，您可以选择将您的对象存储与计算资源联合托管在一个位置，从而提供尽可能高的访问速度。此外，为了进一步提高访问速度并支持每秒数十万个请求，S3 Express One Zone 存储类中的数据存储在新的桶类型中：Amazon S3 目录桶。无论键名或访问模式如何，每个目录存储桶均可支持数十万的每秒事务数（TPS）。

Amazon S3 Express One Zone 存储类设计为在单个可用区内提供 99.95% 的可用性，并由 Amazon S3 服务等级协议提供保障。使用 S3 Express One Zone，您的数据将冗余地存储在单个可用区中的多个设备上。S3 Express One Zone 设计为通过快速检测和修复任何丢失的冗余来处理并发设备故障。在现有设备出现故障时，S3 Express One Zone 会自动将请求转移到相同可用区内的新设备。这种冗余有助于确保不间断地访问可用区中的数据。

S3 Express One Zone 非常适合任何需要尽可能减少访问对象所需延迟的应用程序。此类应用程序可以是人机交互式工作流，例如视频编辑，在这种情况下，创意专业人士需要在通过用户界面访问内容时获得快速的响应。对数据响应能力具有类似要求的分析和机器学习工作负载也可以从 S3 Express One Zone 中获益，尤其是大量访问小文件或者有大量随机访问情况时。S3 Express One Zone 可以与其它 AWS 服务一起使用来支持分析和人工智能与机器学习（AI/ML）工作负载，例如 Amazon EMR、Amazon SageMaker 和 Amazon Athena。



使用 S3 Express One Zone 时，您可以使用网关 VPC 端点，与虚拟私有云（VPC）中的目录桶进行交互。借助网关端点，您可以从 VPC 访问 S3 Express One Zone 目录桶，而无需为 VPC 配备互联网网关或 NAT 设备，也无需任何额外费用。

对于目录桶，您可以使用您用于通用桶和其它存储类的相同 Amazon S3 API 操作和特征。这包括适用于 Amazon S3 的 Mountpoint、具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）、S3 批量操作和 S3 屏蔽公共访问权限。您可以通过 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、AWS SDK 和 Amazon S3 REST API 访问 S3 Express One Zone。

有关 S3 Express One Zone 的更多信息，请参阅以下主题。

- 概述
- S3 Express One Zone 的功能
- 相关服务
- 后续步骤

概述

为了优化性能和减少延迟，S3 Express One Zone 引入了以下新概念。

单可用区

Amazon S3 Express One Zone 存储类设计为在单个可用区内提供 99.95% 的可用性，并由 Amazon S3 服务等级协议提供保障。使用 S3 Express One Zone，您的数据将冗余地存储在单个可用区中的多个设备上。S3 Express One Zone 设计为通过快速检测和修复任何丢失的冗余来处理并发设备故障。在现有设备出现故障时，S3 Express One Zone 会自动将请求转移到相同可用区内的新设备。这种冗余有助于确保不间断地访问可用区中的数据。

可用区是 AWS 区域 中一个或多个具有冗余电源、网络和连接的离散数据中心。创建目录存储桶时，您可以选择存储桶所在的可用区以及 AWS 区域。

目录桶

Amazon S3 桶有两种类型：S3 通用桶和 S3 目录桶。通用存储桶是默认的 Amazon S3 存储桶类型，用于绝大多数 S3 应用场景。目录存储桶仅使用 S3 Express One Zone 存储类，该类专为需要稳定的毫秒级延迟的工作负载或注重性能的应用程序而设计。选择最适合您的应用程序和性能要求的桶类型。

目录桶将数据按层次结构组织到目录中，而不是通用桶的扁平存储结构。目录存储桶没有前缀限制，单个目录可以横向扩展。

目录存储桶使用 S3 Express One Zone 存储类，该类专为注重性能的应用程序使用而构建。使用 S3 Express One Zone，您可以选择单个可用区，并可以选择将您的对象存储与计算资源放在同一个位置，从而提供尽可能高的访问速度。这与通用桶不同，后者在 AWS 区域的多个可用区中以冗余方式存储对象。

有关目录存储桶的更多信息，请参阅目录桶。有关通用存储桶的更多信息，请参阅存储桶概述。

端点和网关 VPC 端点

目录桶的桶管理 API 操作可通过区域端点使用，称为区域端点 API 操作。区域端点 API 操作的示例包括 `CreateBucket` 和 `DeleteBucket`。创建目录存储桶后，您可以使用可用区端点 API 操作来上传和管理目录存储桶中的对象。可用区端点 API 操作可通过可用区端点执行。可用区端点 API 操作的示例包括 `PutObject` 和 `CopyObject`。

您可以使用网关 VPC 端点从 VPC 访问 S3 Express One Zone。创建网关端点后，您可以将其添加作为路由表中的目标，用于从您的 VPC 流向 S3 Express One Zone 的流量。与 Amazon S3 一样，使用网关端点不会产生任何额外费用。有关如何配置网关 VPC 端点的更多信息，请参阅 S3 Express One Zone 联网

基于会话的授权

借助 S3 Express One Zone，您可以通过基于会话的新机制，对请求进行身份验证和授权，该机制经过优化，可提供极低的延迟。您可以使用 `CreateSession` 来请求临时凭证，以便提供对存储桶的低延迟访问。这些临时凭证的作用范围限制为特定的 S3 目录存储桶。会话令牌仅用于可用区（对象级）操作（CopyObject 除外）。有关更多信息，请参阅 `CreateSession` 授权。

S3 Express One Zone 支持的 AWS SDK 代表您处理会话的建立和刷新。为了保护您的会话，临时安全凭证在 5 分钟后过期。您在下载和安装 AWS SDK 并配置必要的 AWS Identity and Access Management（IAM）权限后，便可立即开始使用 API 操作。

S3 Express One Zone 的功能

以下 S3 功能可用于 S3 Express One Zone。有关支持的 API 操作和不支持的特征的完整列表，请参阅S3 Express One Zone 有哪些不同？。

访问管理和安全性

对于目录存储桶，您可以使用以下功能来审计和管理访问权限。默认情况下，目录存储桶为私有，只有被明确授予访问权限的用户才可以访问。与可以在存储桶、前缀或对象标签级设置访问控制边界的通用存储桶不同，目录存储桶只能在存储桶级设置访问控制边界。有关更多信息，请参阅 适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

- S3 屏蔽公共访问权限 – 默认情况下，所有 S3 屏蔽公共访问权限设置均在桶级启用。无法修改此默认设置。
- S3 对象所有权（默认情况下为强制桶拥有者）– 目录桶不支持访问控制列表（ACL）。目录桶对于 S3 对象所有权自动使用强制桶拥有者设置。强制桶拥有者意味着 ACL 被禁用，桶拥有者自动拥有并完全控制桶中的每个对象。无法修改此默认设置。
- AWS Identity and Access Management（IAM）– IAM 有助于您安全地控制对目录存储桶的访问权限。您可以使用 IAM，通过 `s3express:CreateSession` 操作授予对桶管理（区域）API 操作和对象管理（可用区）API 操作的访问权限。有关更多信息，请参阅 适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。与对象管理操作不同，存储桶管理操作不能跨账户。只有存储桶拥有者可以执行这些操作。
- 存储桶策略 – 使用基于 IAM 的策略语言，为目录存储桶配置基于资源的权限。您还可以使用 IAM 来控制对 `CreateSession` API 的访问权限，这允许您使用可用区或对象管理 API 操作。您可以向可用区 API 操作授予同账户或跨账户访问权限。有关 S3 Express One Zone 权限和策略的更多信息，请参阅适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

- 适用于 S3 的 IAM Access Analyzer – 评估和监控您的访问策略，确保这些策略仅提供对 S3 资源的预期访问权限。

日记账记录和监控

S3 Express One Zone 使用以下 S3 日志记录和监控工具，您可以使用这些工具来监控和控制资源的使用方式：

- Amazon CloudWatch 指标 – 通过使用 CloudWatch 收集和跟踪指标，监控您的 AWS 资源和应用程序。S3 Express One Zone 使用与其他 Amazon S3 存储类相同的 CloudWatch 命名空间（`AWS/S3`），并支持目录存储桶的日常存储指标：`BucketSizeBytes` 和 `NumberOfObjects`。有关更多信息，请参阅 使用 Amazon CloudWatch 监控指标。
- AWS CloudTrail 日志 – AWS CloudTrail 是一项 AWS 服务，通过记录用户、角色或 AWS 服务执行的操作，协助您实现运营和风险审计、治理以及 AWS 账户合规性。对于 S3 Express One Zone，CloudTrail 将区域端点 API 操作（例如 `CreateBucket` 和 `PutBucketPolicy`）捕获为管理事件，并将可用区 API 操作（例如 `GetObject` 和 `PutObject`）捕获为数据事件。这些事件包括了在 AWS Management Console、AWS Command Line Interface（AWS CLI）、AWS SDK 和 AWS API 操作中执行的操作。有关更多信息，请参阅使用 AWS CloudTrail 为 S3 Express One Zone 记录日志。

> **Note**
>
> S3 Express One Zone 不支持 Amazon S3 服务器访问日志。

对象管理

创建目录桶后，您可以使用 Amazon S3 控制台、AWS SDK 和 AWS CLI 管理对象存储。以下特征可用于 S3 Express One Zone 的对象管理：

- S3 批量操作 – 使用批量操作对目录存储桶中的对象执行批量操作，例如复制和调用 AWS Lambda 函数。例如，您可以使用批量操作在目录存储桶和通用存储桶之间复制对象。通过批量操作，您可以使用 AWS SDK 或 AWS CLI，或者只需在 Amazon S3 控制台中单击几次，即可通过单个 S3 请求大规模管理数十亿个对象。
- 导入 – 创建目录存储桶后，您可以使用 Amazon S3 控制台中的导入功能在存储桶中填充对象。导入是一种创建批量操作作业的简化方法，用于将对象从通用存储桶复制到目录存储桶。

AWS SDK 和客户端库

创建目录桶并将对象上传到桶后，您可以使用以下方法管理对象存储。

- 适用于 Amazon S3 的 Mountpoint – 适用于 Amazon S3 的 Mountpoint 是一种开源文件客户端，可提供高吞吐量的访问，从而降低 Amazon S3 上数据湖的计算成本。适用于 Amazon S3 的 Mountpoint 将本地文件系统 API 调用转换为 S3 对象 API 调用，如 `GET` 和 `LIST`。它非常适合有大量读取操作的数据湖工作负载，在这种情况下需要处理数 PB 的数据，且需要 Amazon S3 提供的高弹性吞吐量来跨数千个实例进行纵向扩展和缩减。
- S3A – S3A 是推荐使用的 Hadoop 兼容接口，用于访问 Amazon S3 中的数据存储。S3A 取代了 S3N Hadoop 文件系统客户端。
- AWS 上的 PyTorch – AWS 上的 PyTorch 是一个开源深度学习框架，它简化了机器学习模型开发以及将其部署到生产环境的过程。
- AWS SDK – 为 Amazon S3 开发应用程序时，您可以使用 AWS SDK。AWS SDK 包装了底层 Amazon S3 REST API，可以简化您的编程任务。有关将 AWS SDK 与 S3 Express One Zone 结合使用的更多信息，请参阅the section called “AWS SDK”。

数据保护和加密

存储在目录桶中的对象自动通过采用 Amazon S3 托管式密钥的服务器端加密（SSE-S3）进行加密。目录桶不支持采用 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、采用客户提供的加密密钥的服务器端加密（SSE-C）或采用 AWS KMS keys 的双层服务器端加密（DSSE-KMS）。有关更多信息，请参阅数据保护和加密 和使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

S3 Express One Zone 可让您选择用于在上传或下载过程中验证数据的校验和算法。您可以选择以下安全哈希算法（SHA）或循环冗余校验（CRC）数据完整性检查算法之一：CRC32、CRC32C、SHA-1 和 SHA-256。S3 Express One Zone 存储类不支持基于 MD5 的校验和。

有关更多信息，请参阅 其他 S3 校验和最佳实践。

AWS 签名版本 4（SigV4）

S3 Express One Zone 使用 AWS 签名版本 4（SigV4）。SigV4 是一种签名协议，用于对通过 HTTPS 发送到 Amazon S3 的请求进行身份验证。S3 Express One Zone 使用 AWS Sigv4 对请求签名。有关更多信息，请参阅《Amazon Simple Storage Service API》参考中的验证请求 (AWS Signature Version 4)。

强一致性

在所有 AWS 区域中，S3 Express One Zone 为针对目录桶中对象的 PUT 和 DELETE 请求，提供了可靠的先写后读一致性。有关更多信息，请参阅 Amazon S3 数据一致性模型。

相关服务

您可以将以下 AWS 服务与 S3 Express One Zone 存储类配合使用，以支持您的特定低延迟使用场景。

- Amazon Elastic Compute Cloud（Amazon EC2）– Amazon EC2 在 AWS Cloud 中提供安全可扩展的计算容量。使用 Amazon EC2 可减少前期的硬件投入，因此您能够快速开发和部署应用程序。您可以使用 Amazon EC2 启动所需数量的虚拟服务器，配置安全性和联网以及管理存储。
- AWS Lambda – Lambda 是一项计算服务，可使您无需预置或管理服务器即可运行代码。您在存储桶上配置通知设置，并向 Amazon S3 授予权限来根据函数的基于资源的权限策略调用函数。
- Amazon Elastic Kubernetes Service（Amazon EKS）– Amazon EKS 是一项托管式服务，无需在 AWS 上安装、操作和维护自己的 Kubernetes 控制面板。Kubernetes 是一个开源系统，用于自动管理、扩展和部署容器化应用程序。
- Amazon Elastic Container Service（Amazon ECS）– Amazon ECS 是完全托管的容器编排服务，可协助您轻松地部署、管理和扩展容器化应用程序。
- Amazon Athena – Athena 是一种交互式查询服务，方便通过使用标准 SQL 直接分析 Amazon S3 中的数据。还可以使用 Athena，通过 Apache Spark 以交互方式运行数据分析，而无需规划、配置或管理资源。在 Athena 上运行 Apache Spark 应用程序时，您需要提交 Spark 代码以供处理并直接接收结果。
- Amazon SageMaker Runtime 模型训练 – Amazon SageMaker Runtime 是一项完全托管式机器学习服务。借助 SageMaker Runtime，数据科学家和开发人员可以快速、轻松地构建和训练机器学习模型，然后直接将模型部署到生产就绪托管环境中。
- AWS Glue – AWS Glue 是一项无服务器数据集成服务，可让使用分析功能的用户轻松发现、准备、移动和集成来自多个来源的数据。您可以使用 AWS Glue 进行分析、机器学习和应用程序开发。AWS Glue 还包括用于编写、运行任务和实施业务工作流程的额外生产率和数据操作工具。
- Amazon EMR – Amazon EMR 是一个托管式集群平台，可简化在 AWS 上运行大数据框架（如 Apache Hadoop 和 Apache Spark）来处理和分析海量数据的过程。

后续步骤

有关使用 S3 Express One Zone 存储类和目录存储桶的更多信息，请参阅以下主题：

- S3 Express One Zone 有哪些不同？
- 教程：开始使用 S3 Express One Zone
- S3 Express One Zone 联网
- 目录桶
- 使用目录桶中的对象
- S3 Express One Zone 的安全性
- 优化 Amazon S3 Express One Zone 性能
- 使用 S3 Express One Zone 进行开发

S3 Express One Zone 有哪些不同？

Amazon S3 Express One Zone 是高性能的单区 Amazon S3 存储类，专门用于为延迟要求极高的应用程序提供稳定的毫秒级数据访问。S3 Express One Zone 是第一种可以在其中选择单个可用区的 S3 存储类，您可以选择将您的对象存储与计算资源联合托管在一个位置，从而提供尽可能高的访问速度。此外，为了进一步提高访问速度并支持每秒数十万个请求，S3 Express One Zone 数据存储在新的存储桶类型中：Amazon S3 目录存储桶。

有关更多信息，请参阅什么是 S3 Express One Zone？ 和目录桶。

您可以使用 Amazon S3 API 创建目录存储桶并访问 S3 Express One Zone 中的数据。Amazon S3 API 与 S3 Express One Zone 和目录存储桶兼容，但存在一些明显的区别。有关 S3 Express One Zone 不同之处的更多信息，请参阅以下主题。

主题

- S3 Express One Zone 的不同之处
- S3 Express One Zone 支持的 API 操作
- S3 Express One Zone 不支持的 Amazon S3 功能

S3 Express One Zone 的不同之处

- 支持的存储桶类型 – S3 Express One Zone 存储类中的对象只能存储在目录存储桶中。有关更多信息，请参阅 目录桶。
- 持久性 – 使用 S3 Express One Zone 时，您的数据将冗余地存储在单个可用区中的多个设备上。S3 Express One Zone 设计为在单个可用区内提供 99.95% 的可用性，并由 Amazon S3 服务等级协议提供支持。有关更多信息，请参阅 单可用区。
- **`ListObjectsV2`** 行为
  - 对于目录存储桶，`ListObjectsV2` 不按字母表顺序返回对象。此外，前缀必须以分隔符结尾，并且只能将“/”指定为分隔符。
  - 对于目录存储桶，`ListObjectsV2` 响应包含仅与正在进行的分段上传相关的前缀。
- 删除行为 – 当您删除目录存储桶中的对象时，Amazon S3 会递归删除对象路径中的所有空目录。例如，如果您删除对象键 `dir1/dir2/file1.txt`，Amazon S3 会删除 `file1.txt`。如果 `dir1/` 和 `dir2/` 目录为空且不包含其他对象，Amazon S3 还会删除这些目录。
- ETag 与校验和 – S3 Express One Zone 的实体标签（ETag）是随机的字母数字字符串，而不是 MD5 校验和。有关在 S3 Express One Zone 中使用其他校验和的更多信息，请参阅 其他 S3 校验和最佳实践。
- **`DeleteObjects`** 请求中的对象键
  - `DeleteObjects` 请求中的对象键必须至少包含一个非空格字符。`DeleteObjects` 请求中不支持完全由空格组成的字符串。
  - `DeleteObjects` 请求中的对象键不能包含 Unicode 控制字符，但换行符（`\n`）、制表符（`\t`）和回车符（`\r`）字符除外。
- 区域端点和可用区端点 – 使用 S3 Express One Zone 时，您必须在所有客户端请求中指定区域。对于区域端点，您可以指定区域，例如 `s3express-control.us-west-2.amazonaws.com`。对于可用区端点，您需要同时指定区域和可用区，例如 `s3express-usw2-az1.us-west-2.amazonaws.com`。有关更多信息，请参阅 区域端点和可用区端点。
- 分段上传 – 与存储在 Amazon S3 中的其他对象一样，您可以使用分段上传流程，上传和复制存储在 S3 Express One Zone 存储类中的大型对象。但是，对存储在 S3 Express One Zone 中的对象使用分段上传流程时，会有以下一些区别。有关更多信息，请参阅 the section called “对目录桶使用分段上传”。
  - 对象的创建日期是分段上传完成的日期。
  - 分段上传的分段编号必须连续。如果您尝试使用非连续分段编号完成分段上传请求，Amazon S3 会生成 HTTP `400 (Bad Request)` 错误。

  - 只有在通过 `s3express:CreateSession` 权限明确向分段上传的发起者授予了对 `AbortMultipartUpload` 的访问权限时，才能中止分段上传请求。有关更多信息，请参阅 适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。
- 清空目录存储桶：通过 AWS Command Line Interface（CLI）执行的 `s3 rm` 命令、通过 Mountpoint 执行的 `delete` 操作以及通过 AWS Management Console执行的清空存储桶选项按钮无法删除目录存储桶中正在进行的分段上传。要删除这些正在进行的分段上传，请使用 `ListMultipartUploads` 操作列出存储桶中正在进行的分段上传，然后使用 `AbortMultupartUpload` 操作中止所有正在进行的分段上传。

S3 Express One Zone 支持的 API 操作

Amazon S3 Express One Zone 存储类同时支持区域（存储桶级或控制面板）和可用区（对象级或数据面板）端点 API 操作。有关更多信息，请参阅S3 Express One Zone 联网 和端点和网关 VPC 端点。

区域端点 API 操作

S3 Express One Zone 支持以下区域端点 API 操作：

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- PutBucketPolicy

可用区端点 API 操作

S3 Express One Zone 支持以下可用区端点 API 操作：

- CreateSession
- CopyObject
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes

- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject
- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload
- ListMultipartUploads
- ListParts
- UploadPart
- UploadPartCopy

S3 Express One Zone 不支持的 Amazon S3 功能

S3 Express One Zone 不支持以下 Amazon S3 功能：

- AWS 托管策略
- 适用于 S3 的 AWS PrivateLink
- MD5 校验和
- 多重验证 (MFA) 删除
- S3 对象锁定
- 申请方付款
- S3 访问授权
- S3 接入点
- 存储桶标签
- Amazon CloudWatch 请求指标
- S3 事件通知
- S3 生命周期
- S3 多区域接入点
- S3 对象 Lambda 接入点
- S3 版本控制
- S3 清单

- S3 复制
- 对象标签
- S3 Select
- 服务器访问日志
- 静态网站托管
- S3 Storage Lens 存储统计管理工具
- S3 Storage Lens 组
- S3 Transfer Acceleration
- 具有 AWS Key Management Service（AWS KMS）密钥的双层服务器端加密（DSSE-KMS）
- 具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）
- 具有客户提供密钥的服务器端加密（SSE-C）
- 在 Amazon S3 控制台中创建新存储桶时复制现有存储桶的设置的选项
- 增强拒绝访问（HTTP `403 Forbidden`）错误消息

教程：开始使用 S3 Express One Zone

Amazon S3 Express One Zone 存储类是第一种可以在其中选择单个可用区的 S3 存储类，您可以选择将您的对象存储与计算资源联合托管在一个位置，从而提供尽可能高的访问速度。S3 Express One Zone 中的数据存储在 S3 目录存储桶中。有关目录存储桶的更多信息，请参阅目录存储桶。

S3 Express One Zone 非常适合任何务必尽可能减少请求延迟的应用程序。此类应用程序可以是人机交互式工作流，例如视频编辑，在这种情况下，创意专业人士需要在通过用户界面访问内容时获得快速的响应。对数据响应能力具有类似要求的分析和机器学习工作负载也可以从 S3 Express One Zone 中获益，尤其是对于有大量访问小文件或大量随机访问的工作负载。S3 Express One Zone 可以与其它 AWS 服务一起使用来支持分析以及人工智能与机器学习（AI/ML）工作负载，例如 Amazon EMR、Amazon Athena、AWS Glue Data Catalog 和 Amazon SageMaker Model Training。可以通过 Amazon S3 控制台、AWS SDK、AWS 命令行界面（AWS CLI）和 Amazon S3 REST API 使用 S3 Express One Zone 存储类和目录存储桶。有关更多信息，请参阅什么是 S3 Express One Zone？和 S3 Express One Zone 有哪些不同？
这是 S3 Express One Zone 工作流程图。

目标

在本教程中，您将学习如何创建网关端点、创建和附加 IAM 策略、创建目录存储桶，然后使用“导入”操作用当前存储在通用存储桶中的对象填充目录存储桶。或者，您可以手动将对象上传到目录存储桶。

主题

先决条件

在开始本教程之前，您必须具有 AWS 账户，您可使用具有正确权限的 AWS Identity and Access Management（IAM）用户登录此账户。

分步

- 创建 AWS 账户
- 在 AWS 账户中创建 IAM 用户（控制台）
- 创建 IAM 策略并将其附加到 IAM 用户或角色（控制台）

创建 AWS 账户

要完成此教程，您需要一个AWS 账户。在注册 AWS 时，将在 AWS 中为您的 AWS 账户 自动注册所有服务，包括 Amazon S3。您只需为使用的服务付费。有关定价的更多信息，请参阅 S3 定价。

在 AWS 账户中创建 IAM 用户（控制台）

AWS Identity and Access Management（IAM）是一项 AWS 服务，有助于管理员安全地控制对 AWS 资源的访问。IAM 管理员可控制哪些用户能够通过身份验证（登录）和获得授权（拥有权限），来在 S3 Express One Zone 中访问对象和使用目录存储桶。使用 IAM 不会产生额外的费用。

默认情况下，用户没有访问目录存储桶和执行 S3 Express One Zone 操作的权限。要授予对目录存储桶和 S3 Express One Zone 操作的访问权限，您可以使用 IAM 创建用户或角色，然后将权限附加到这些身份。有关如何创建 IAM 用户的更多信息，请参阅 《IAM 用户指南》中的创建 IAM 用户（控制台）。有关如何创建 IAM 角色的更多信息，请参阅《IAM 用户指南》中的创建向 IAM 用户委派权限的角色。

为简单起见，本教程创建并使用 IAM 用户。完成本教程后，请记住 删除 IAM 用户。对于生产用途，我们建议您遵循《IAM 用户指南》中的 IAM 中的安全最佳实践。最佳实践要求人类用户将联合身份验证与身份提供商结合使用，以使用临时凭证访问 AWS。另一项最佳实践是要求工作负载使用带有 IAM 角色的临时凭证访问 AWS。要详细了解如何使用 AWS IAM Identity Center 创建具有临时凭证的用户，请参阅《AWS IAM Identity Center 用户指南》中的 Getting started。

> **Warning**
>
> IAM 用户具有长期凭证，这会带来安全风险。为帮助减轻这种风险，我们建议仅向这些用户提供执行任务所需的权限，并在不再需要这些用户时将其移除。

创建 IAM 策略并将其附加到 IAM 用户或角色（控制台）

默认情况下，用户没有目录存储桶和 S3 Express One Zone 操作的权限。要授予对目录存储桶的访问权限，您可以使用 IAM 创建用户、组或角色，然后将权限附加到这些身份。对于 S3 Express One Zone 访问权限，目录存储桶是唯一可以包括在存储桶策略或 IAM 身份策略中的资源。

要在 S3 Express One Zone 中使用区域端点 API 操作（存储桶级或控制面板操作），请使用 IAM 授权模型，该模型不涉及会话管理。对于操作，将单独授予权限。要使用可用区端点 API 操作（对象级或数据面板操作），可以使用 `CreateSession` 来创建和管理会话，这些会话经过优化，可为数据请求提供低延迟授权。要检索和使用会话令牌，您必须在基于身份的策略或存储桶策略中，允许对目录存储桶执行 `s3express:CreateSession` 操作。如果您在 Amazon S3 控制台中通过 AWS 命令行界面（AWS CLI）或使用 AWS SDK 访问 S3 Express One Zone，S3 Express One Zone 会代表您创建会话。有关更多信息，请参阅 `CreateSession` 授权和适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

创建 IAM 策略并将该策略附加到 IAM 用户（或角色）

1. 登录 AWS 管理控制台，并打开 IAM 管理控制台。
2. 在导航窗格中，选择策略。
3. 选择创建策略。
4. 选择 JSON。
5. 将以下策略复制到策略编辑器窗口。在创建目录存储桶或使用 S3 Express One Zone 之前，您必须向 AWS Identity and Access Management（IAM）角色或用户授予必要的权限。此示例策略支持访问 `CreateSession` API 操作（与其它可用区或对象级 API 操作结合使用）和所有区域端点（存储桶级）API 操作。此策略允许将 `CreateSession` API 操作用于所有目录存储桶，

但仅允许将区域端点 API 操作用于指定的目录存储桶。要使用此示例策略，请将 *user input placeholders* 替换为您自己的信息。

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "AllowAccessRegionalEndpointAPIs",
             "Effect": "Allow",
             "Action": [
                 "s3express:DeleteBucket",
                 "s3express:DeleteBucketPolicy",
                 "s3express:CreateBucket",
                 "s3express:PutBucketPolicy",
                 "s3express:GetBucketPolicy",
                 "s3express:ListAllMyDirectoryBuckets"
             ],

             "Resource": "arn:aws:s3express:region:account_id:bucket/bucket-base-name--azid--x-s3/*"
         },
         {
             "Sid": "AllowCreateSession",
             "Effect": "Allow",
             "Action": "s3express:CreateSession",
             "Resource": "*"
         }
     ]
 }
```

6. 选择下一步。
7. 为该策略命名。

> **Note**
>
> S3 Express One Zone 不支持存储桶标签。

8. 选择创建策略。
9. 创建 IAM 策略后，可以将其附加到 IAM 用户。在导航窗格中，选择策略。
10. 在搜索栏中，输入策略的名称。
11. 从操作菜单中，选择附加。

12. 在按实体类型筛选下，选择 IAM 用户或角色。
13. 在搜索字段中，键入您要使用的用户或角色的名称。
14. 选择附加策略。

步骤 1：配置网关 VPC 端点

您可以通过网关虚拟私有云（VPC）端点访问可用区和区域 API 操作。网关端点可让流量在不通过 NAT 网关的情况下到达 S3 Express One Zone。我们强烈建议使用网关端点，因为在使用 S3 Express One Zone 时，此类端点可提供最佳的联网路径。您可以从 VPC 访问 S3 Express One Zone 目录存储桶，而无需为 VPC 配备互联网网关或 NAT 设备，也无需任何额外费用。使用以下过程可配置连接到 S3 Express One Zone 存储类对象和目录存储桶的网关端点。

要访问 S3 Express One Zone，您需要使用不同于标准 Amazon S3 端点的区域和可用区端点。根据您使用的 Amazon S3 API 操作，需要区域端点或可用区端点。有关各端点类型支持的 API 操作的完整列表，请参阅 S3 Express One Zone 支持的 API 操作。您必须通过网关虚拟私有云（VPC）端点访问可用区端点和区域端点。

使用以下过程创建连接到 S3 Express One Zone 存储类对象和目录存储桶的网关端点。

配置网关 VPC 端点

1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 在侧导航窗格中的虚拟私有云下，选择端点。
3. 选择创建端点。
4. 为端点创建名称。
5. 对于 Service category（服务类别），选择 AWS 服务。
6. 在服务下，使用筛选条件 Type=Gateway 进行搜索，然后选择 com.amazonaws.***region***.s3express 旁边的选项按钮。
7. 对于 VPC，选择要在其中创建端点的 VPC。
8. 对于 Route tables（路由表），选择端点要使用的路由表。Amazon VPC 自动添加路由，将流向服务的流量指向端点网络接口。
9. 对于策略，请选择完整访问权限以允许所有主体通过 VPC 端点对所有资源执行所有操作。否则，选择自定义以附加 VPC 端点策略，该策略控制主体通过 VPC 端点对资源执行操作的权限。
10. 选择创建端点。

创建网关端点后，您可以使用区域 API 端点和可用区 API 端点来访问 Amazon S3 Express One Zone 存储类对象和目录存储桶。

步骤 2：创建目录存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要在其中创建存储桶的区域。

   > **Note**
   >
   > 要最大程度地减少延迟和成本以及满足法规要求，请选择一个靠近您的区域。在某一区域存储的对象将一直留在该区域，除非您特意将其转移到其他区域。有关 Amazon S3 AWS 区域的列表，请参阅《Amazon Web Services 一般参考》中的 AWS 服务 端点。

3. 在左侧导航窗格中，选择存储桶。
4. 选择创建存储桶。

   此时将打开创建存储桶页面。
5. 在常规配置下，查看将在其中创建存储桶的 AWS 区域。
6. 在存储桶类型下，请选择目录。

   > **Note**
   >
   > - 如果您选择的区域不支持目录桶，则桶类型选项将消失，桶类型默认为通用桶。要创建目录桶，您必须选择受支持的区域。有关支持目录桶和 Amazon S3 Express One Zone 存储类的区域列表，请参阅the section called “S3 Express One Zone 可用区和区域”。
   > - 在创建存储桶后，便无法更改存储桶类型。

   对于可用区，请选择计算服务本地的可用区。有关支持目录桶和 S3 Express One Zone 存储类的可用区列表，请参阅the section called “S3 Express One Zone 可用区和区域”。

> **Note**
>
> 创建存储桶后无法更改可用区。

7. 在可用区下，选中复选框以确认在可用区中断时，您的数据可能不可用或丢失。

> **Important**
>
> 尽管目录桶存储在单个可用区中的多个设备上，但目录桶不会跨可用区冗余存储数据。

8. 对于存储桶名称，请输入目录存储桶的名称。

   以下命名规则适用于目录存储桶。

   - 在选定的 AWS 区域和可用区内是唯一的。
   - 名称的长度必须介于 3（最小）到 63（最大）个字符之间，包括后缀。
   - 仅包含小写字母、数字和连字符（-）。
   - 以字母或数字开头和结尾。
   - 必须包含以下后缀：`--azid--x-s3`。
   - 存储桶名称不得以前缀 `xn--` 开头。
   - 存储桶名称不得以前缀 `sthree-` 开头。
   - 存储桶名称不得以前缀 `sthree-configurator` 开头。
   - 存储桶名称不得以前缀 `amzn-s3-demo-` 开头。
   - 存储桶名称不得以后缀 `-s3alias` 结尾。此后缀是为接入点别名预留的。有关更多信息，请参阅 为您的 S3 存储桶接入点使用存储桶式别名。
   - 存储桶名称不得以后缀 `--ol-s3` 结尾。此后缀是为对象 Lambda 接入点别名预留的。有关更多信息，请参阅 如何为您的 S3 存储桶对象 Lambda 接入点使用存储桶式别名。
   - 存储桶名称不得以后缀 `.mrap` 结尾。此后缀预留用于多区域接入点名称。有关更多信息，请参阅 命名 Amazon S3 多区域接入点的规则。

   后缀将自动添加到您使用控制台创建目录存储桶时提供的基本名称中。此后缀包括您选择的可用区的可用区 ID。

   创建存储桶后，便无法更改其名称。有关给存储桶命名的更多信息，请参阅存储桶命名规则。

> **Important**
>
> 请勿在桶名称中包含敏感信息，如账号。存储桶名称会显示在指向存储桶中的对象的 URL 中。

9. 在对象所有权下，将自动启用强制桶拥有者设置，并禁用所有访问控制列表（ACL）。对于目录存储桶，无法启用 ACL。

   已禁用 ACL

   - 强制存储桶拥有者（默认）– ACL 已禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响对 S3 存储桶中数据的访问权限。存储桶专门使用策略来定义访问控制。

     Amazon S3 中的大多数现代使用案例不再需要使用 ACL。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。 。

10. 在此桶的屏蔽公共访问权限设置下，目录桶的所有屏蔽公共访问权限设置已自动启用。无法修改目录桶的这些设置。有关阻止公共访问的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

11. 在服务器端加密设置下，Amazon S3 应用采用 Amazon S3 托管式密钥的服务器端加密（SSE-S3），作为对所有 Amazon S3 存储桶的基本加密级别。上传到目录桶的所有对象都使用 SSE-S3 进行加密。对于目录桶，无法修改加密类型。有关 SSE-S3 的更多信息，请参阅the section called “Amazon S3 托管式加密密钥（SSE-S3）”。

12. 请选择创建存储桶。

    创建存储桶后，您可以向存储桶中添加文件和文件夹。有关更多信息，请参阅 the section called “使用目录桶中的对象”。

以下步骤演示如何使用 Amazon S3 控制台中的“导入”操作在目录存储桶中填充数据。

步骤 3：将数据导入到目录存储桶

要完成此步骤，您必须有一个包含对象的通用存储桶，并且它与目录存储桶位于同一个 AWS 区域中。

在 Amazon S3 中创建目录存储桶后，可以使用 Amazon S3 控制台中的“导入”操作在新的存储桶中填充数据。导入过程可让您选择要从中导入数据的前缀或通用存储桶，而无需单独指定要复制的所有对象，从而简化了将数据复制到目录存储桶的过程。导入过程使用 S3 批量操作，这会复制所选前缀或通用存储桶中的对象。可以通过 S3 批量操作任务详细信息页面监控“导入”复制任务的进度。

使用“导入”操作

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择与目录存储桶所在的可用区关联的区域。
3. 在左侧导航窗格中，选择存储桶，然后选择目录存储桶选项卡。选择要将对象导入到的目录存储桶。
4. 选择导入。
5. 对于源，输入包含要导入的对象的通用存储桶（或包含前缀的存储桶路径）。要从列表中选择现有的通用存储桶，请选择浏览 S3。
6. 在权限部分，可以选择自动生成 IAM 角色。或者，可以从列表中选择 IAM 角色，或直接输入 IAM 角色 ARN。

   - 要允许 Amazon S3 代表您创建新的 IAM 角色，请选择创建新的 IAM 角色。

     > **Note**
     >
     > 如果您的源对象使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）进行加密，请不要选择创建新的 IAM 角色选项。而是指定具有 `kms:Decrypt` 权限的现有 IAM 角色。
     > Amazon S3 将使用此权限来解密您的对象。在导入过程中，Amazon S3 将使用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）对这些对象进行重新加密。

   - 要从列表中选择现有 IAM 角色，请选择从现有 IAM 角色中选择。
   - 要通过输入 Amazon 资源名称（ARN）指定现有 IAM 角色，请选择输入 IAM 角色 ARN，然后在相应字段中输入 ARN。
7. 查看目标和复制的对象设置部分中显示的信息。如果目标部分中的信息正确，请选择导入以启动复制作业。

   Amazon S3 控制台在批量操作页面上显示新作业的状态。有关作业的更多信息，请选择作业名称旁边的选项按钮，然后在操作菜单上选择查看详细信息。要打开要导入对象的目录存储桶，请选择查看导入目标。

步骤 4：手动将对象上传到目录存储桶

也可以手动将对象上传到目录存储桶。

手动上传对象

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面右上角的导航栏中，选择当前所显示的 AWS 区域的名称。接下来，选择与目录存储桶所在的可用区关联的区域。
3. 在左侧导航窗格中，选择存储桶。
4. 选择目录桶选项卡。
5. 选择要将文件夹或文件上传到的桶的名称。

   > **Note**
   >
   > 如果您选择的目录存储桶与本教程前面的步骤中使用的目录存储桶相同，则目录存储桶将包含从“导入”工具上传的对象。请注意，这些对象现在存储在 S3 Express One Zone 存储类中。

6. 在对象列表中，选择上传。
7. 在上传页面上，执行以下操作之一：
   - 将文件和文件夹拖放到虚线上传区域。
   - 选择添加文件或添加文件夹，选择要上传的文件或文件夹，然后选择打开或上传。
8. 在校验和下，选择要使用的校验和函数。

   > **Note**
   >
   > 我们建议使用 CRC32 和 CRC32C，以便在 S3 Express One Zone 存储类上获得最佳性能。有关更多信息，请参阅其它 S3 校验和最佳实践。

   （可选）如果您要上传大小不到 16 MB 的单个对象，也可以指定预先计算的校验和值。当您提供预先计算的值时，Amazon S3 会将该值与它使用所选校验和函数计算的值进行比较。如果值不匹配，则上传不会开始。

9. 权限和属性部分中的选项会自动设置为默认设置，无法修改。将自动启用“屏蔽公共访问权限”，而无法为目录桶启用 S3 版本控制和 S3 对象锁定。

   （可选）如果要以键值对的形式向对象添加元数据，请展开属性部分，然后在元数据部分中选择添加元数据。

10. 要上传列出的文件和文件夹，请选择上传。

    Amazon S3 会上传您的对象和文件夹。上传完成后，您可以在上传：状态页面上看到成功消息。

    您已成功创建了目录存储桶并将对象上传到存储桶。

步骤 5：清空目录存储桶

可以使用 Amazon S3 控制台来清空 Amazon S3 目录存储桶。

清空目录桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面右上角的导航栏中，选择当前所显示的 AWS 区域的名称。接下来，选择与目录存储桶所在的可用区关联的区域。
3. 在左侧导航窗格中，选择存储桶。
4. 选择目录桶选项卡。
5. 选择您要清空的桶名称旁边的选项按钮，然后选择清空。
6. 在清空存储桶页面上，通过在文本字段中输入 **`permanently delete`** 来确认要清空存储桶，然后选择清空。
7. 在清空桶：状态页面上监控桶清空过程的进度。

步骤 6：删除目录存储桶

清空目录存储桶并中止所有正在进行的分段上传后，可以使用 Amazon S3 控制台删除存储桶。

删除目录存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 在页面右上角的导航栏中，选择当前所显示的 AWS 区域的名称。接下来，选择与目录存储桶所在的可用区关联的区域。
3. 在左侧导航窗格中，选择存储桶。
4. 选择目录桶选项卡。
5. 在目录桶列表中，选择要删除的桶旁边的选项按钮。
6. 选择删除。
7. 在删除桶页面上，在文本字段中输入桶的名称以确认删除您的桶。

> **Important**
>
> 无法撤消删除目录存储桶的操作。

8. 要删除您的目录存储桶，请选择删除存储桶。

后续步骤

在本教程中，您学习了如何创建目录存储桶和使用 S3 Express One Zone 存储类。完成本教程后，您可以探索要与 S3 Express One Zone 存储类一起使用的相关 AWS 服务。

您可以将以下 AWS 服务与 S3 Express One Zone 存储类配合使用，以支持您的特定低延迟使用场景。

- Amazon Elastic Compute Cloud（Amazon EC2）– Amazon EC2 在 AWS Cloud 中提供安全可扩展的计算容量。使用 Amazon EC2 可减少前期的硬件投入，因此您能够快速开发和部署应用程序。您可以使用 Amazon EC2 启动所需数量的虚拟服务器，配置安全性和联网以及管理存储。
- AWS Lambda – Lambda 是一项计算服务，可使您无需预置或管理服务器即可运行代码。您在存储桶上配置通知设置，并向 Amazon S3 授予权限来根据函数的基于资源的权限策略调用函数。
- Amazon Elastic Kubernetes Service（Amazon EKS）– Amazon EKS 是一项托管式服务，无需在 AWS 上安装、操作和维护自己的 Kubernetes 控制面板。Kubernetes 是一个开源系统，用于自动管理、扩展和部署容器化应用程序。
- Amazon Elastic Container Service（Amazon ECS）– Amazon ECS 是完全托管的容器编排服务，可协助您轻松地部署、管理和扩展容器化应用程序。
- Amazon EMR – Amazon EMR 是一个托管式集群平台，可简化在 AWS 上运行大数据框架（如 Apache Hadoop 和 Apache Spark）来处理和分析海量数据的过程。
- Amazon Athena – Athena 是一种交互式查询服务，方便通过使用标准 SQL 直接分析 Amazon S3 中的数据。还可以使用 Athena，通过 Apache Spark 以交互方式运行数据分析，而无需规划、配置或

管理资源。在 Athena 上运行 Apache Spark 应用程序时，您需要提交 Spark 代码以供处理并直接接收结果。

- AWS Glue Data Catalog – AWS Glue 是一项无服务器数据集成服务，可让使用分析功能的用户轻松发现、准备、移动和集成来自多个来源的数据。您可以使用 AWS Glue 进行分析、机器学习和应用程序开发。AWSGlue Data Catalog 是一个集中式存储库，用于存储有关贵组织数据集的元数据。它充当数据来源的位置、架构和运行时指标的索引。
- Amazon SageMaker Runtime 模型训练 – Amazon SageMaker Runtime 是一项完全托管式机器学习服务。借助 SageMaker Runtime，数据科学家和开发人员可以快速、轻松地构建和训练机器学习模型，然后直接将模型部署到生产就绪托管环境中。

有关 S3 Express One Zone 的更多信息，请参阅什么是 S3 Express One Zone？和 S3 Express One Zone 有哪些不同？

S3 Express One Zone 联网

要访问 Amazon S3 Express One Zone 存储类对象和目录存储桶，您需要使用不同于标准 Amazon S3 端点的区域和可用区 API 端点。根据您使用的 S3 API 操作，需要区域端点或可用区端点。有关按端点类型统计的 API 操作的完整列表，请参阅 S3 Express One Zone 支持的 API 操作。

您可以通过网关虚拟私有云（VPC）端点访问可用区和区域 API 操作。要配置网关 VPC 端点，请参阅the section called “配置 VPC 网关端点”。

以下主题介绍使用网关 VPC 端点访问 S3 Express One Zone 时的联网要求。

主题

- 端点
- 配置 VPC 网关端点

端点

您可以使用网关 VPC 端点，从 VPC 访问 Amazon S3 Express One Zone 存储类对象和目录存储桶。S3 Express One Zone 使用区域和可用区 API 端点。根据您使用的 Amazon S3 API 操作，需要区域端点或可用区端点。使用网关端点不会发生任何额外费用。

存储桶级（或控制面板）API 操作可通过区域端点使用，称为区域端点 API 操作。区域端点 API 操作的示例包括 `CreateBucket` 和 `DeleteBucket`。在创建目录存储桶时，您需要选择一个用于创建目

录存储桶的可用区。创建目录存储桶后，您可以使用可用区端点 API 操作来上传和管理目录存储桶中的对象。

对象级（或数据面板）API 操作可通过可用区端点使用，称为可用区端点 API 操作。可用区端点 API 操作的示例包括 `CreateSession` 和 `PutObject`。

下表显示了在各个区域和可用区中可以使用的区域和可用区 API 端点。

配置 VPC 网关端点

使用以下过程创建连接到 Amazon S3 Express One Zone 存储类对象和目录存储桶的网关端点。

配置网关 VPC 端点

1. 打开位于 https://console.aws.amazon.com/vpc/ 的 Amazon VPC 控制台。
2. 在导航窗格中，选择端点。
3. 选择 创建端点。
4. 为端点创建名称。
5. 对于 Service category（服务类别），选择 AWS 服务。
6. 对于服务，添加筛选条件 Type=Gateway，然后选择 com.amazonaws.***region***.s3express 旁边的选项按钮。
7. 对于 VPC，选择要在其中创建端点的 VPC。
8. 对于 Route tables（路由表），选择端点要使用的路由表。Amazon VPC 自动添加路由，将流向服务的流量指向端点网络接口。
9. 对于策略，请选择完整访问权限以允许所有主体通过 VPC 端点对所有资源执行所有操作。否则，选择自定义以附加 VPC 端点策略，该策略控制主体通过 VPC 端点对资源执行操作的权限。
10. （可选）要添加标签，请选择添加新标签，然后输入标签键和标签值。
11. 选择创建端点。

创建网关端点后，您可以使用区域 API 端点和可用区 API 端点来访问 Amazon S3 Express One Zone 存储类对象和目录存储桶。

目录桶

Amazon S3 存储桶有两种类型：通用存储桶和目录存储桶。选择最适合您的应用程序和性能要求的桶类型：

- 通用存储桶是最初的 S3 存储桶类型，建议用于大多数使用案例和访问模式。通用存储桶还允许跨所有存储类（S3 Express One Zone 除外）存储对象。
- 目录存储桶使用 S3 Express One Zone 存储类，如果您的应用程序注重性能，并且受益于个位数毫秒的 PUT 和 GET 延迟，则建议使用该存储类。

目录存储桶用于需要一致的个位数毫秒延迟的工作负载或性能关键型应用程序。目录桶将数据按层次结构组织到目录中，而不是通用桶的扁平存储结构。目录存储桶没有前缀限制，单个目录可以横向扩展。

目录存储桶使用 S3 Express One Zone 存储类，该存储类可在单个可用区内的多个设备上存储数据，但不会跨可用区冗余存储数据。创建目录桶时，我们建议您指定 Amazon EC2、Amazon Elastic Kubernetes Service 或 Amazon Elastic Container Service（Amazon ECS）计算实例的本地 AWS 区域和可用区，以优化性能。

每个 AWS 账户中最多可以创建 10 个目录桶，并且对桶中可以存储的对象数量没有限制。您的存储桶限额适用于您的 AWS 账户中的每个区域。如果您的应用程序需要提高此限制，请联系 AWS Support。有关更多信息，请访问服务限额控制台。

> **Important**
>
> 在至少 90 天期间内没有请求活动的目录桶将转换为非活动状态。处于非活动状态时，目录存储桶暂时无法进行读取和写入。非活动存储桶会保留所有存储、对象元数据和存储桶元数据。现有的存储费用适用于非活动桶。如果您向非活动桶发出访问请求，该桶通常会在几分钟内转换为活动状态。在此转换期间，读取和写入操作会返回 HTTP 503 (Service Unavailable) 错误代码。

以下主题提供有关目录存储桶的信息。有关通用存储桶的更多信息，请参阅存储桶概述。

主题

- 创建目录存储桶
- 查看目录存储桶属性
- 管理目录存储桶的存储桶策略
- 清空目录存储桶
- 删除目录存储桶
- 列出目录存储桶
- 将 HeadBucket 与目录桶结合使用

可用区

创建目录存储桶时，您可以选择可用区和 AWS 区域。

目录存储桶使用 S3 Express One Zone 存储类，该存储类专用于注重性能的应用程序。S3 Express One Zone 是第一种可以在其中选择单个可用区的 S3 存储类，您可以选择将您的对象存储与计算资源联合托管在一个位置，从而提供尽可能高的访问速度。

使用 S3 Express One Zone，您的数据将冗余地存储在单个可用区中的多个设备上。S3 Express One Zone 设计为在单个可用区内提供 99.95% 的可用性，并由 Amazon S3 服务等级协议提供支持。有关更多信息，请参阅 单可用区

目录存储桶名称

目录存储桶名称由您提供的基本名称和包含存储桶所在可用区 ID 的后缀组成。目录存储桶名称必须使用以下格式并遵循目录存储桶的命名规则：

```
bucket-base-name--azid--x-s3
```

例如，以下目录存储桶名称包含可用区 ID `usw2-az1`：

```
bucket-base-name--usw2-az1--x-s3
```

有关更多信息，请参阅 目录存储桶命名规则。

目录

目录桶将数据按层次结构组织到目录中，而不是通用桶的扁平存储结构。每个 S3 目录存储桶均可支持数十万个每秒事务数（TPS），与存储桶内的目录数量无关。

对于分层命名空间，对象键中的分隔符很重要。唯一支持的分隔符是正斜杠（/）。目录由分隔符边界确定。例如，对象键 dir1/dir2/file1.txt 会自动创建目录 dir1/ 和 dir2/，并将对象 file1.txt 添加到路径 dir1/dir2/file1.txt 中的 /dir2 目录。

目录存储桶索引模型会为 ListObjectsV2 API 操作返回未排序的结果。如果您需要将结果限制在存储桶的某个子部分，则可以在 prefix 参数中指定子目录路径，例如 prefix=dir1/。

键名称

对于目录存储桶，多个对象键共用的子目录将使用第一个对象键创建。同一子目录的其它对象键使用之前创建的子目录。此模型使您可以灵活地选择最适合应用程序的对象键，同时同样支持稀疏目录和密集目录。

访问管理

目录存储桶默认在存储桶级别启用了所有 S3 屏蔽公共访问权限设置。S3 对象所有权设置为强制存储桶拥有者，并禁用访问控制列表（ACL）。无法修改这些设置。

默认情况下，用户没有目录存储桶和 S3 Express One Zone 操作的权限。要授予对目录存储桶的访问权限，您可以使用 IAM 创建用户、组或角色，然后将权限附加到这些身份。有关更多信息，请参阅适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

使用目录存储桶

有关使用目录存储桶的更多信息，请参阅以下主题。

主题

- 目录存储桶命名规则
- 创建目录存储桶
- 查看目录存储桶属性
- 管理目录存储桶的存储桶策略
- 清空目录存储桶
- 删除目录存储桶
- 列出目录存储桶
- 将 HeadBucket 与目录桶结合使用

目录存储桶命名规则

在 Amazon S3 中创建目录存储桶时，以下存储桶命名规则适用。有关通用存储桶命名规则，请参阅存储桶命名规则。

目录存储桶名称由您提供的基本名称和包含桶所在 AWS 可用区 ID 的后缀以及 `--x-s3` 组成。

```
base-name--azid--x-s3
```

例如，以下目录存储桶名称包含可用区 ID `usw2-az1`：

```
bucket-base-name--usw2-az1--x-s3
```

> **Note**
>
> 当您使用控制台创建目录存储桶时，系统会自动将后缀添加到您提供的基本名称。此后缀包括您选择的可用区的可用区 ID。
> 当您使用 API 创建目录存储桶时，必须在请求中提供包括可用区 ID 在内的完整后缀。有关可用区 ID 的列表，请参阅 S3 Express One Zone 可用区和区域。

以下命名规则适用于目录存储桶。

- 在选定的 AWS 区域和可用区内是唯一的。
- 名称的长度必须介于 3（最小）到 63（最大）个字符之间，包括后缀。
- 仅包含小写字母、数字和连字符（-）。
- 以字母或数字开头和结尾。
- 必须包含以下后缀：`--azid--x-s3`。
- 存储桶名称不得以前缀 `xn--` 开头。
- 存储桶名称不得以前缀 `sthree-` 开头。
- 存储桶名称不得以前缀 `sthree-configurator` 开头。
- 存储桶名称不得以前缀 `amzn-s3-demo-` 开头。
- 存储桶名称不得以后缀 `-s3alias` 结尾。此后缀是为接入点别名预留的。有关更多信息，请参阅 为您的 S3 存储桶接入点使用存储桶式别名。
- 存储桶名称不得以后缀 `--ol-s3` 结尾。此后缀是为对象 Lambda 接入点别名预留的。有关更多信息，请参阅 如何为您的 S3 存储桶对象 Lambda 接入点使用存储桶式别名。

- 存储桶名称不得以后缀 .mrap 结尾。此后缀预留用于多区域接入点名称。有关更多信息，请参阅 命名 Amazon S3 多区域接入点的规则。

创建目录存储桶

要开始使用 Amazon S3 Express One Zone 存储类，您需要创建一个目录存储桶。S3 Express One Zone 存储类只能用于目录存储桶。S3 Express One Zone 存储类支持低延迟使用案例，并在单个可用区内提供更快的数据处理速度。如果您的应用程序注重性能，并且受益于个位数毫秒的 PUT 和 GET 延迟，我们建议您创建一个目录存储桶，以便可以使用 S3 Express One Zone 存储类。

Amazon S3 存储桶有两种类型：通用存储桶和目录存储桶。您应该选择适合您应用程序和性能要求的存储桶类型。通用存储桶是最初的 S3 存储桶类型。对于大多数使用案例和访问模式，建议使用通用桶，通用桶允许在除 S3 Express One Zone 之外的所有存储类中存储对象。有关通用存储桶的更多信息，请参阅存储桶概述。

目录存储桶使用 S3 Express One Zone 存储类，该存储类专用于需要一致的个位数毫秒延迟的工作负载或性能关键型应用程序。S3 Express One Zone 是第一种可以在其中选择单个可用区的 S3 存储类，您可以选择将您的对象存储与计算资源联合托管在一个位置，从而提供尽可能高的访问速度。创建目录存储桶时，您可以选择指定 Amazon EC2、Amazon Elastic Kubernetes Service 或 Amazon Elastic Container Service（Amazon ECS）计算实例的本地 AWS 区域和可用区，以优化性能。

使用 S3 Express One Zone，您的数据将冗余地存储在单个可用区中的多个设备上。S3 Express One Zone 设计为在单个可用区内提供 99.95% 的可用性，并由 Amazon S3 服务等级协议提供支持。有关更多信息，请参阅 单可用区

目录桶将数据按层次结构组织到目录中，而不是通用桶的扁平存储结构。目录存储桶没有前缀限制，单个目录可以横向扩展。

有关目录存储桶的更多信息，请参阅目录桶。

目录存储桶名称

目录存储桶名称必须遵循以下格式并遵守目录存储桶命名规则：

```
bucket-base-name--azid--x-s3
```

例如，以下目录存储桶名称包含可用区 ID usw2-az1：

```
bucket-base-name--usw2-az1--x-s3
```

有关目录存储桶命名规则的更多信息，请参阅目录存储桶命名规则。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要在其中创建存储桶的区域。

   > **Note**
   >
   > 要最大程度地减少延迟和成本以及满足法规要求，请选择一个靠近您的区域。在某一区域存储的对象将一直留在该区域，除非您特意将其转移到其他区域。有关 Amazon S3 AWS 区域的列表，请参阅《Amazon Web Services 一般参考》中的 AWS 服务 端点。

3. 在左侧导航窗格中，选择存储桶。
4. 选择创建存储桶。

   此时将打开创建存储桶页面。
5. 在常规配置下，查看将在其中创建存储桶的 AWS 区域。
6. 在存储桶类型下，请选择目录。

   > **Note**
   >
   > - 如果您选择的区域不支持目录桶，则桶类型选项将消失，桶类型默认为通用桶。要创建目录桶，您必须选择受支持的区域。有关支持目录桶和 Amazon S3 Express One Zone 存储类的区域列表，请参阅the section called “S3 Express One Zone 可用区和区域”。
   > - 在创建存储桶后，便无法更改存储桶类型。

   对于可用区，请选择计算服务本地的可用区。有关支持目录桶和 S3 Express One Zone 存储类的可用区列表，请参阅the section called “S3 Express One Zone 可用区和区域”。

   > **Note**
   >
   > 创建存储桶后无法更改可用区。

7. 在可用区下，选中复选框以确认在可用区中断时，您的数据可能不可用或丢失。

> **Important**
>
> 尽管目录桶存储在单个可用区中的多个设备上，但目录桶不会跨可用区冗余存储数据。

8. 对于存储桶名称，请输入目录存储桶的名称。

   以下命名规则适用于目录存储桶。

   - 在选定的 AWS 区域和可用区内是唯一的。
   - 名称的长度必须介于 3（最小）到 63（最大）个字符之间，包括后缀。
   - 仅包含小写字母、数字和连字符（-）。
   - 以字母或数字开头和结尾。
   - 必须包含以下后缀：`--`*`azid`*`--x-s3`。
   - 存储桶名称不得以前缀 `xn--` 开头。
   - 存储桶名称不得以前缀 `sthree-` 开头。
   - 存储桶名称不得以前缀 `sthree-configurator` 开头。
   - 存储桶名称不得以前缀 `amzn-s3-demo-` 开头。
   - 存储桶名称不得以后缀 `-s3alias` 结尾。此后缀是为接入点别名预留的。有关更多信息，请参阅 为您的 S3 存储桶接入点使用存储桶式别名。
   - 存储桶名称不得以后缀 `--ol-s3` 结尾。此后缀是为对象 Lambda 接入点别名预留的。有关更多信息，请参阅 如何为您的 S3 存储桶对象 Lambda 接入点使用存储桶式别名。
   - 存储桶名称不得以后缀 `.mrap` 结尾。此后缀预留用于多区域接入点名称。有关更多信息，请参阅 命名 Amazon S3 多区域接入点的规则。

   后缀将自动添加到您使用控制台创建目录存储桶时提供的基本名称中。此后缀包括您选择的可用区的可用区 ID。

   创建存储桶后，便无法更改其名称。有关给存储桶命名的更多信息，请参阅存储桶命名规则。

> **Important**
>
> 请勿在桶名称中包含敏感信息，如账号。存储桶名称会显示在指向存储桶中的对象的 URL 中。

9. 在对象所有权下，将自动启用强制桶拥有者设置，并禁用所有访问控制列表（ACL）。对于目录存储桶，无法启用 ACL。

   已禁用 ACL

   - 强制存储桶拥有者（默认）– ACL 已禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响对 S3 存储桶中数据的访问权限。存储桶专门使用策略来定义访问控制。

     Amazon S3 中的大多数现代使用案例不再需要使用 ACL。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。 。

10. 在此桶的屏蔽公共访问权限设置下，目录桶的所有屏蔽公共访问权限设置已自动启用。无法修改目录桶的这些设置。有关阻止公共访问的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。
11. 在服务器端加密设置下，Amazon S3 应用采用 Amazon S3 托管式密钥的服务器端加密（SSE-S3），作为对所有 Amazon S3 存储桶的基本加密级别。上传到目录桶的所有对象都使用 SSE-S3 进行加密。对于目录桶，无法修改加密类型。有关 SSE-S3 的更多信息，请参阅the section called “Amazon S3 托管式加密密钥（SSE-S3）”。
12. 请选择创建存储桶。

    创建存储桶后，您可以向存储桶中添加文件和文件夹。有关更多信息，请参阅 the section called “使用目录桶中的对象”。

使用 AWS SDK

SDK for Go

此示例说明如何使用 AWS SDK for Go 创建目录桶。

Example

```
var bucket = "..."

func runCreateBucket(c *s3.Client) {
    resp, err := c.CreateBucket(context.Background(), &s3.CreateBucketInput{
        Bucket: &bucket,
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            Location: &types.LocationInfo{
                Name: aws.String("usw2-az1"),
                Type: types.LocationTypeAvailabilityZone,
            },
```

```
            Bucket: &types.BucketInfo{
                DataRedundancy: types.DataRedundancySingleAvailabilityZone,
                Type:           types.BucketTypeDirectory,
            },
        },
    })
    var terr *types.BucketAlreadyOwnedByYou
    if errors.As(err, &terr) {
        fmt.Printf("BucketAlreadyOwnedByYou: %s\n", aws.ToString(terr.Message))
        fmt.Printf("noop...\n")
        return
    }
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("bucket created at %s\n", aws.ToString(resp.Location))
}
```

SDK for Java 2.x

此示例说明如何使用 AWS SDK for Java 2.x 创建目录桶。

Example

```
public static void createBucket(S3Client s3Client, String bucketName) {

    //Bucket name format is {base-bucket-name}--{az-id}--x-s3
    //example: doc-example-bucket--usw2-az1--x-s3 is a valid name for a directory
 bucket created in
    //Region us-west-2, Availability Zone 2

    CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
             .location(LocationInfo.builder()
                     .type(LocationType.AVAILABILITY_ZONE)
                     .name("usw2-az1").build()) //this must match the Region and
 Availability Zone in your bucket name
             .bucket(BucketInfo.builder()
                    .type(BucketType.DIRECTORY)
                    .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                    .build()).build();
    try {
```

```
            CreateBucketRequest bucketRequest =
CreateBucketRequest.builder().bucket(bucketName).createBucketConfiguration(bucketConfigurat
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            System.out.println(response);
    }

    catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
         }
    }
```

AWS SDK for JavaScript

此示例说明如何使用 AWS SDK for JavaScript 创建目录桶。

Example

```
// file.mjs, run with Node.js v16 or higher
// To use with the preview build, place this in a folder
// inside the preview build directory, such as /aws-sdk-js-v3/workspace/

import { S3 } from "@aws-sdk/client-s3";

const region = "us-east-1";
const zone = "use1-az4";
const suffix = `${zone}--x-s3`;

const s3 = new S3({ region });

const bucketName = `...--${suffix}`;

const createResponse = await s3.createBucket(
    { Bucket: bucketName,
      CreateBucketConfiguration: {Location: {Type: "AvailabilityZone", Name: zone},
      Bucket: { Type: "Directory", DataRedundancy: "SingleAvailabilityZone" }}
    }
   );
```

AWS SDK for .NET

此示例说明如何使用 AWS SDK for .NET 创建目录桶。

Example

```
using (var amazonS3Client = new AmazonS3Client())
{
    var putBucketResponse = await amazonS3Client.PutBucketAsync(new PutBucketRequest
    {

       BucketName = "DOC-EXAMPLE-BUCKET--usw2-az1--x-s3",
       PutBucketConfiguration = new PutBucketConfiguration
       {
         BucketInfo = new BucketInfo { DataRedundancy =
 DataRedundancy.SingleAvailabilityZone, Type = BucketType.Directory },
         Location = new LocationInfo { Name = "usw2-az1", Type =
 LocationType.AvailabilityZone }
       }
     }).ConfigureAwait(false);
}
```

SDK for PHP

此示例说明如何使用 AWS SDK for PHP 创建目录桶。

Example

```
require 'vendor/autoload.php';

$s3Client = new S3Client([

    'region'      => 'us-east-1',
]);


$result = $s3Client->createBucket([
    'Bucket' => 'doc-example-bucket--use1-az4--x-s3',
    'CreateBucketConfiguration' => [
        'Location' => ['Name'=> 'use1-az4', 'Type'=> 'AvailabilityZone'],
        'Bucket' => ["DataRedundancy" => "SingleAvailabilityZone" ,"Type" =>
 "Directory"]   ],
]);
```

SDK for Python

此示例说明如何使用 AWS SDK for Python (Boto3) 创建目录桶。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def create_bucket(s3_client, bucket_name, availability_zone):
    '''
    Create a directory bucket in a specified Availability Zone

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to create; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :param availability_zone: String; Availability Zone ID to create the bucket in,
 for example, 'usw2-az1'
    :return: True if bucket is created, else False
    '''

    try:
        bucket_config = {
                'Location': {
                    'Type': 'AvailabilityZone',
                    'Name': availability_zone
                },
                'Bucket': {
                    'Type': 'Directory',
                    'DataRedundancy': 'SingleAvailabilityZone'
                }
            }
        s3_client.create_bucket(
            Bucket = bucket_name,
            CreateBucketConfiguration = bucket_config
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    availability_zone = 'usw2-az1'
    s3_client = boto3.client('s3', region_name = region)
```

```
    create_bucket(s3_client, bucket_name, availability_zone)
```

SDK for Ruby

此示例说明如何使用 AWS SDK for Ruby 创建目录桶。

Example

```
s3 = Aws::S3::Client.new(region:'us-west-2')
s3.create_bucket(
  bucket: "bucket_base_name--az_id--x-s3",
  create_bucket_configuration: {
    location: { name: 'usw2-az1', type: 'AvailabilityZone' },
    bucket: { data_redundancy: 'SingleAvailabilityZone', type: 'Directory' }
  }
)
```

使用 AWS CLI

此示例说明如何使用 AWS CLI 创建目录桶。要使用该命令，请将*#######* 替换为您自己的信息。

创建目录存储桶时，必须提供配置详细信息并使用以下命名约定：*bucket-base-name*--*azid*--x-s3

```
aws s3api create-bucket
--bucket bucket-base-name--azid--x-s3
--create-bucket-configuration 'Location={Type=AvailabilityZone,Name=usw2-
az1},Bucket={DataRedundancy=SingleAvailabilityZone,Type=Directory}'
--region us-west-2
```

有关更多信息，请参阅 AWS Command Line Interface 中的 create-bucket。

查看目录存储桶属性

您可以使用 Amazon S3 控制台查看和配置 Amazon S3 目录桶的属性。有关更多信息，请参阅 目录桶和 什么是 S3 Express One Zone？。

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 在左侧导航窗格中，选择存储桶。
3. 选择目录桶选项卡。
4. 在目录存储桶列表中，选择要查看其属性的存储桶的名称。
5. 选择属性选项卡。
6. 在属性选项卡上，您可以查看桶的以下属性：

   - 目录桶概述 – 您可以查看桶的 AWS 区域、可用区、Amazon 资源名称（ARN）和创建日期。
   - 默认加密 – Amazon S3 应用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3），作为对所有 S3 存储桶的基本加密级别。对于目录存储桶，无法修改此设置。Amazon S3 会在将对象保存到磁盘之前对其进行加密，并在下载对象时对其进行解密。有关更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

     有关目录存储桶支持的功能的更多信息，请参阅S3 Express One Zone 的功能。

管理目录存储桶的存储桶策略

您可以使用 Amazon S3 控制台和 AWS SDK 添加、删除、更新和查看 Amazon S3 目录桶的桶策略。有关更多信息，请参阅以下主题。有关 S3 Express One Zone 支持的 AWS Identity and Access Management（IAM）操作和条件键的更多信息，请参阅适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。有关目录存储桶的存储桶策略示例，请参阅S3 Express One Zone 目录存储桶策略示例。

主题

- 添加存储桶策略
- 查看存储桶策略
- 删除存储桶策略

添加存储桶策略

要向目录存储桶添加存储桶策略，您可以使用 Amazon S3 控制台、AWS SDK 或 AWS CLI。

使用 S3 控制台

创建或编辑存储桶策略

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 在左侧导航窗格中，选择存储桶。
3. 选择目录桶选项卡。
4. 在目录存储桶列表中，选择要将文件夹和文件上传到的存储桶的名称。
5. 选择 Permissions（权限）选项卡。
6. 在 Bucket policy（存储桶策略）下，请选择 Edit（编辑）。将出现 Edit bucket policy（编辑存储桶策略）页面。
7. 要自动生成策略，请选择策略生成器。

   如果选择 Policy generator（策略生成器），AWS 策略生成器将在新窗口中打开。

   如果您不想使用 AWS 策略生成器，则可以在策略部分添加或编辑 JSON 语句。

   a. 在 AWS 策略生成器页面上，对于选择策略类型，选择 S3 存储桶策略。
   b. 通过在提供的字段中输入信息来添加语句，然后选择添加语句。对您要添加的所需数量的语句重复此步骤。有关这些字段的更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

      > **Note**
      >
      > 为方便起见，编辑桶策略页面会在策略文本字段上方显示当前桶的桶 ARN（Amazon 资源名称）。您可以复制此 ARN，以便在 AWS 策略生成器页面上的语句中使用。

   c. 添加完语句后，请选择生成策略。
   d. 复制生成的策略文本，请选择 Close（关闭），然后返回到 Amazon S3 控制台中的 Edit bucket policy（编辑存储桶策略）页面。
8. 在 Policy（策略）框中，编辑现有策略或从 AWS 策略生成器粘贴存储桶策略。确保在保存策略之前解决安全警告、错误、一般警告和建议。

   > **Note**
   >
   > 存储桶策略的大小限制为 20 KB。

9. 选择保存更改，此操作将让您返回到权限选项卡。

使用 AWS SDK

SDK for Java 2.x

Example

PutBucketPolicy AWS SDK for Java 2.x

```
public static void setBucketPolicy(S3Client s3Client, String bucketName, String
 policyText) {

       //sample policy text
       /**
        * policy_statement = {
         *         'Version': '2012-10-17',
         *         'Statement': [
         *             {
         *                 'Sid': 'AdminPolicy',
         *                 'Effect': 'Allow',
         *                 'Principal': {
         *                     "AWS": "111122223333"
         *                 },
         *                 'Action': 's3express:*',
         *                 'Resource':
 'arn:aws:s3express:region:111122223333:bucket/bucket-base-name--azid--x-s3'
         *             }
         *         ]
         *     }
         */
         System.out.println("Setting policy:");
         System.out.println("----");
         System.out.println(policyText);
         System.out.println("----");
         System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

         try {
             PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                     .bucket(bucketName)
                     .policy(policyText)
                     .build();
             s3Client.putBucketPolicy(policyReq);
             System.out.println("Done!");
         }
```

```
        catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

使用 AWS CLI

此示例显示如何使用 AWS CLI 将存储桶策略添加到目录存储桶。要使用该命令，请将####### 替换为您自己的信息。

```
aws s3api put-bucket-policy --bucket bucket-base-name--azid--x-s3 --policy file://
bucket_policy.json
```

bucket_policy.json：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AdminPolicy",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express*",
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/"
        }
    ]
}
```

有关更多信息，请参阅 AWS Command Line Interface 中的 put-bucket-policy。

查看存储桶策略

要查看目录存储桶的存储桶策略，请使用以下示例。

使用 AWS CLI

此示例说明如何使用 AWS CLI 查看附加到目录存储桶的存储桶策略。要使用该命令，请将*#######*替换为您自己的信息。

```
aws s3api get-bucket-policy --bucket bucket-base-name--azid--x-s3
```

有关更多信息，请参阅 AWS Command Line Interface 中的 get-bucket-policy。

删除存储桶策略

要删除目录存储桶的存储桶策略，请使用以下示例。

使用 AWS SDK

SDK for Java 2.x

Example

`DeleteBucketPolicy` AWS SDK for Java 2.x

```
public static void deleteBucketPolicy(S3Client s3Client, String bucketName) {
      try {
          DeleteBucketPolicyRequest deleteBucketPolicyRequest =
 DeleteBucketPolicyRequest
                  .builder()
                  .bucket(bucketName)
                  .build()
          s3Client.deleteBucketPolicy(deleteBucketPolicyRequest);
          System.out.println("Successfully deleted bucket policy");
      }

      catch (S3Exception e) {
          System.err.println(e.awsErrorDetails().errorMessage());
          System.exit(1);
      }
```

使用 AWS CLI

此示例显示如何使用 AWS CLI 删除目录存储桶的存储桶策略。要使用该命令，请将*#######* 替换为您自己的信息。

```
aws s3api delete-bucket-policy --bucket bucket-base-name--azid--x-s3
```

有关更多信息，请参阅 AWS Command Line Interface 中的 delete-bucket-policy。

清空目录存储桶

您可以使用 Amazon S3 控制台清空 Amazon S3 目录桶。有关目录存储桶的更多信息，请参阅目录桶。

清空目录存储桶之前，请注意以下几点：

- 清空目录存储桶时，您将删除所有对象，但会保留目录存储桶。
- 清空目录桶后，清空操作无法撤销。
- 当正在执行清空桶操作时添加到目录桶的对象可能被删除。

如果您还要删除桶，请注意以下几点：

- 必须先删除目录存储桶中的所有对象，然后才能删除目录存储桶本身。
- 必须先中止目录存储桶中正在进行的分段上传，然后才能删除存储桶本身。

> **Note**
>
> 通过 AWS Command Line Interface（CLI）执行的 `s3 rm` 命令、通过 Mountpoint 执行的 `delete` 操作以及通过 AWS Management Console执行的清空存储桶选项按钮无法删除目录存储桶中正在进行的分段上传。要删除这些正在进行的分段上传，请使用 `ListMultipartUploads` 操作列出存储桶中正在进行的分段上传，然后使用 `AbortMultipartUpload` 操作中止所有正在进行的分段上传。

要删除目录存储桶，请参阅删除目录存储桶。要中止正在进行的分段上传，请参阅the section called “中止分段上传”。

要清空通用存储桶，请参阅清空存储桶。

使用 S3 控制台

清空目录桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择目录桶选项卡。
4. 选择您要清空的桶名称旁边的选项按钮，然后选择清空。
5. 在清空存储桶页面上，通过在文本字段中输入 **`permanently delete`** 来确认要清空存储桶，然后选择清空。
6. 在清空桶：状态页面上监控桶清空过程的进度。

删除目录存储桶

您只能删除空的 Amazon S3 目录桶。在删除目录桶之前，必须删除桶中的所有对象并中止所有正在进行的分段上传。

要清空目录存储桶，请参阅清空目录存储桶。要中止正在进行的分段上传，请参阅the section called “中止分段上传”。

要删除通用存储桶，请参阅删除存储桶。

使用 S3 控制台

清空目录桶并中止所有正在进行的分段上传后，您可以删除桶。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择目录桶选项卡。
4. 在目录桶列表中，选择要删除的桶旁边的选项按钮。
5. 选择 删除。
6. 在删除桶页面上，在文本字段中输入桶的名称以确认删除您的桶。

> **Important**
>
> 无法撤消删除目录存储桶的操作。

7. 要删除您的目录存储桶，请选择删除存储桶。

使用 AWS SDK

以下示例使用 AWS SDK for Java 2.x 和 AWS SDK for Python (Boto3) 删除目录桶。

SDK for Java 2.x

Example

```
public static void deleteBucket(S3Client s3Client, String bucketName) {

    try {
        DeleteBucketRequest del = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(del);
        System.out.println("Bucket " + bucketName + " has been deleted");
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def delete_bucket(s3_client, bucket_name):
    '''
    Delete a directory bucket in a specified Region
```

```
    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to delete; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :return: True if bucket is deleted, else False
    '''

    try:
        s3_client.delete_bucket(Bucket = bucket_name)
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
```

使用 AWS CLI

此示例说明如何使用 AWS CLI 删除目录存储桶。要使用该命令，请将####### 替换为您自己的信息。

```
aws s3api delete-bucket --bucket bucket-base-name--azid--x-s3 --region us-west-2
```

有关更多信息，请参阅 AWS Command Line Interface 中的 delete-bucket。

列出目录存储桶

以下示例显示如何使用 AWS SDK 和 AWS CLI 列出目录存储桶。

使用 AWS SDK

SDK for Java 2.x

Example

下面的示例通过使用 AWS SDK for Java 2.x 列出目录桶。

```
 public static void listBuckets(S3Client s3Client) {
```

```
        try {
            ListDirectoryBucketsRequest listDirectoryBucketsRequest =
 ListDirectoryBucketsRequest.builder().build();
            ListDirectoryBucketsResponse response =
 s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);
            if (response.hasBuckets()) {
                for (Bucket bucket: response.buckets()) {
                    System.out.println(bucket.name());
                    System.out.println(bucket.creationDate());
                }
            }
        }

        catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

SDK for Python

Example

下面的示例通过使用 AWS SDK for Python (Boto3) 列出目录桶。

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_directory_buckets(s3_client):
'''
Prints a list of all directory buckets in a Region

:param s3_client: boto3 S3 client
:return: True if there are buckets in the Region, else False
'''
try:
    response = s3_client.list_directory_buckets()
    for bucket in response['Buckets']:
        print (bucket['Name'])
except ClientError as e:
    logging.error(e)
    return False
```

```
return True


if __name__ == '__main__':
    region = 'us-east-1'
    s3_client = boto3.client('s3', region_name = region)
    list_directory_buckets(s3_client)
```

AWS SDK for .NET

Example

下面的示例通过使用 AWS SDK for .NET 列出目录桶。

```
var listDirectoryBuckets = await amazonS3Client.ListDirectoryBucketsAsync(new
 ListDirectoryBucketsRequest
{
  MaxDirectoryBuckets = 10
  }).ConfigureAwait(false);
```

SDK for PHP

Example

下面的示例通过使用 AWS SDK for PHP 列出目录桶。

```
require 'vendor/autoload.php';

$s3Client = new S3Client([
    'region'      => 'us-east-1',
]);
$result = $s3Client->listDirectoryBuckets();
```

SDK for Ruby

Example

下面的示例通过使用 AWS SDK for Ruby 列出目录桶。

```
s3 = Aws::S3::Client.new(region:'us-west-1')
```

```
s3.list_directory_buckets
```

使用 AWS CLI

以下 `list-directory-buckets` 示例命令显示如何使用 AWS CLI 列出 *us-east-1* 区域中的目录存储桶。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3api list-directory-buckets --region us-east-1
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 list-directory-buckets。

将 HeadBucket 与目录桶结合使用

以下 AWS SDK 示例显示如何使用 HeadBucket API 操作来确定 Amazon S3 目录桶是否存在以及您是否有权访问它。

使用 AWS SDK

以下 AWS SDK for Java 2.x 示例说明如何确定桶是否存在以及您是否有权访问该桶。

SDK for Java 2.x

Example

AWS SDK for Java 2.x

```
public static void headBucket(S3Client s3Client, String bucketName) {
   try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest
                .builder()
                .bucket(bucketName)
                .build();
        s3Client.headBucket(headBucketRequest);
        System.out.format("Amazon S3 bucket: \"%s\" found.", bucketName);
   }

   catch (S3Exception e) {
       System.err.println(e.awsErrorDetails().errorMessage());
       System.exit(1);
   }
```

```
}
```

使用 AWS CLI

以下 head-bucket 示例命令显示了如何使用 AWS CLI 来确定目录存储桶是否存在以及您是否有权访问该存储桶。要运行此命令，请将用户输入占位符替换为您自己的信息。

```
aws s3api head-bucket --bucket bucket-base-name--azid--x-s3
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 head-bucket。

使用目录桶中的对象

创建 Amazon S3 目录桶后，您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK 处理对象。

有关对存储在 S3 Express One Zone 存储类中的对象进行批量对象操作的更多信息，请参阅对象管理。有关在目录桶中导入、上传、复制、删除和下载对象以及读取对象中元数据的更多信息，请参阅下面的主题。

主题

- 将对象导入到目录存储桶
- 对 S3 Express One Zone 使用批量操作
- 将对象上传到目录存储桶
- 对目录桶使用分段上传
- 将对象复制到目录存储桶
- 删除目录存储桶中的对象
- 下载目录桶中的对象
- 将 HeadObject 与目录桶结合使用

将对象导入到目录存储桶

在 Amazon S3 中创建目录存储桶后，您可以使用导入操作在新的存储桶中填充数据。导入是一种创建 S3 批量操作作业的简化方法，用于将对象从通用存储桶复制到目录存储桶。

> **Note**
>
> 以下限制适用于导入作业：
>
> - 源存储桶和目标存储桶必须位于相同的 AWS 区域和账户中。
> - 源存储桶不能是目录存储桶。
> - 不支持大于 5 GB 的对象，复制操作中将省略这些对象。
> - Glacier Flexible Retrieval、Glacier Deep Archive、Intelligent-Tiering 归档访问层和 Intelligent-Tiering 深度归档层存储类中的对象必须先恢复，然后才能导入。
> - 使用 MD5 校验和算法导入的对象将转换为使用 CRC32 校验和。
> - 导入的对象采用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）。
> - 导入的对象使用 Express One Zone 存储类，其定价结构与通用桶使用的存储类不同。在导入大量对象时，请考虑这种成本差异。

配置导入作业时，您需要指定从中复制现有对象的源存储桶或前缀。您还可以提供一个 AWS Identity and Access Management（IAM）角色，该角色拥有访问源对象的权限。然后，Amazon S3 会启动批量操作作业，复制对象并自动应用适当的存储类和校验和设置。

要配置导入作业，您可以使用 Amazon S3 控制台。

使用 Amazon S3 控制台

将对象导入到目录存储桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶，然后选择目录存储桶选项卡。选择要将对象导入到的目录存储桶旁边的选项按钮。
3. 选择导入。
4. 对于源，输入包含要导入的对象的通用存储桶（或包含前缀的存储桶路径）。要从列表中选择现有的通用存储桶，请选择浏览 S3。
5. 要获得访问和复制源对象的权限，请执行以下任一操作来指定具有导入源对象所需权限的 IAM 角色：

   - 要允许 Amazon S3 代表您创建新的 IAM 角色，请选择创建新的 IAM 角色。

> **Note**
>
> 如果您的源对象使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）进行加密，请不要选择创建新的 IAM 角色选项。而是指定具有 `kms:Decrypt` 权限的现有 IAM 角色。
> Amazon S3 将使用此权限来解密您的对象。在导入过程中，Amazon S3 将使用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）对这些对象进行重新加密。

- 要从列表中选择现有 IAM 角色，请选择从现有 IAM 角色中选择。
- 要通过输入 Amazon 资源名称（ARN）指定现有 IAM 角色，请选择输入 IAM 角色 ARN，然后在相应字段中输入 ARN。

6. 查看目标和复制的对象设置部分中显示的信息。如果目标部分中的信息正确，请选择导入以启动复制作业。

   Amazon S3 控制台在批量操作页面上显示新作业的状态。有关作业的更多信息，请选择作业名称旁边的选项按钮，然后在操作菜单上选择查看详细信息。要打开要导入对象的目录存储桶，请选择查看导入目标。

对 S3 Express One Zone 使用批量操作

您可以使用 Amazon S3 批量操作对存储在 S3 存储桶中的对象执行操作。要了解有关 S3 批量操作的更多信息，请参阅对 Amazon S3 对象执行大规模批量操作。

以下主题讨论对存储在目录桶的 S3 Express One Zone 存储类中的对象执行批量操作。

主题

- 对目录存储桶使用批量操作
- 主要区别

对目录存储桶使用批量操作

您可以对存储在目录桶中的对象执行复制操作和调用 AWS Lambda 函数操作。通过复制，您可以在相同类型的桶之间复制对象（例如，从目录桶复制到目录桶）。您也可以在通用存储桶和目录存储桶之间进行复制。通过调用 AWS Lambda 函数操作，您可以通过 Lambda 函数使用您定义的代码对目录存储桶中的对象执行操作。

复制对象

您可以在相同的存储桶类型之间进行复制，也可以在目录存储桶和通用存储桶之间进行复制。当您复制到目录桶时，必须为此桶类型使用正确的 Amazon 资源名称（ARN）格式。目录存储桶的 ARN 格式为 `arn:aws:s3express:region:account-id:bucket/bucket-base-name--x-s3`。

您也可以使用 S3 控制台中的导入操作在目录存储桶中填充数据。导入是一种创建批量操作作业的简化方法，用于将对象从通用存储桶复制到目录存储桶。对于从通用存储桶到目录存储桶的导入复制作业，S3 会自动生成清单。有关更多信息，请参阅将对象导入到目录桶和指定清单。

调用 Lambda 函数（**`LambdaInvoke`**）

使用批量操作来调用处理目录存储桶的 Lambda 函数有特殊要求。例如，您必须使用 v2 JSON 调用架构来构建 Lambda 请求，并在创建任务时指定 InvocationSchemaVersion 2.0。有关更多信息，请参阅调用 AWS Lambda 函数。

主要区别

下面列出了使用批量操作对存储在目录桶（使用 S3 Express One Zone 存储类）中的对象执行批量操作时的主要区别：

- Amazon S3 会自动加密上传到 S3 存储桶的所有新对象。S3 存储桶的默认加密配置始终处于启用状态，并至少设置为具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）。对于目录桶，仅支持 SSE-S3。如果您发出 `CopyObject` 请求来对目录桶（源或目标）设置采用客户提供密钥的服务器端加密（SSE-C）或采用 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS），则响应将返回 HTTP `400 (Bad Request)` 错误。
- 无法标记目录存储桶中的对象。您只能指定一个空标签集。默认情况下，批量操作会复制标签。如果您在通用桶和目录桶之间复制带有标签的对象，则将收到 `501 (Not Implemented)` 响应。
- S3 Express One Zone 可让您选择用于在上传或下载过程中验证数据的校验和算法。您可以选择以下安全哈希算法（SHA）或循环冗余校验（CRC）数据完整性检查算法之一：CRC32、CRC32、SHA-1 和 SHA-256。S3 Express One Zone 存储类不支持基于 MD5 的校验和。
- 默认情况下，所有 Amazon S3 桶都将 S3 对象所有权设置设为强制桶拥有者，并禁用访问控制列表（ACL）。对于目录存储桶，无法修改此设置。您可以将对象从通用存储桶复制到目录存储桶。然而，当您复制到目录桶或从目录桶复制时，无法覆盖默认 ACL。
- 无论您如何指定清单，列表本身都必须存储在通用存储桶中。批量操作无法从目录桶中导入现有清单，也无法将生成的清单保存到目录桶。但是，清单中描述的对象可以存储在目录存储桶中。

- 批量操作无法将目录桶指定为 S3 清单报告中的位置。清单报告不支持目录桶。您可以使用 `ListObjectsV2` API 操作列出对象，为目录桶中的对象创建清单文件。然后，您可以在 CSV 文件中插入该列表。

授予访问权限

要执行复制作业，您必须拥有以下权限：

- 要将对象从一个目录存储桶复制到另一个目录存储桶，您必须拥有 `s3express:CreateSession` 权限。
- 要将对象从目录存储桶复制到通用存储桶，您必须拥有将对象副本写入目标存储桶的 `s3express:CreateSession` 权限和 `s3:PutObject` 权限。
- 要将对象从通用桶复制到目录桶，您必须拥有读取正在复制的源对象的 `s3express:CreateSession` 权限和 `s3:GetObject` 权限。

  有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CopyObject。
- 要调用 Lambda 函数，您必须根据 Lambda 函数授予对资源的权限。要确定需要哪些权限，请检查相应的 API 操作。

将对象上传到目录存储桶

创建 Amazon S3 目录桶后，可以将对象上传到该桶。以下示例显示如何使用 S3 控制台和 AWS SDK 将对象上传到目录桶。有关使用 S3 Express One Zone 进行批量上传对象操作的信息，请参阅对象管理。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择目录桶选项卡。
4. 选择要将文件夹或文件上传到的桶的名称。
5. 在对象列表中，选择上传。
6. 在上传页面上，执行以下操作之一：

   - 将文件和文件夹拖放到虚线上传区域。

- 选择添加文件或添加文件夹，选择要上传的文件或文件夹，然后选择打开或上传。

7. 在校验和下，选择要使用的校验和函数。

   （可选）如果您要上传大小不到 16MB 的单个对象，也可以指定预先计算的校验和值。当您提供预先计算的值时，Amazon S3 会将该值与它使用所选校验和函数计算的值进行比较。如果值不匹配，则上传不会开始。

8. 权限和属性部分中的选项会自动设置为默认设置，无法修改。将自动启用“屏蔽公共访问权限”，而无法为目录桶启用 S3 版本控制和 S3 对象锁定。

   （可选）如果要以键值对的形式向对象添加元数据，请展开属性部分，然后在元数据部分中选择添加元数据。

9. 要上传列出的文件和文件夹，请选择上传。

   Amazon S3 会上传您的对象和文件夹。上传完成后，您可以在上传：状态页面上看到成功消息。

使用 AWS SDK

SDK for Java 2.x

Example

```
public static void putObject(S3Client s3Client, String bucketName, String objectKey,
 Path filePath) {
       //Using File Path to avoid loading the whole file into memory
       try {
           PutObjectRequest putObj = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   //.metadata(metadata)
                   .build();
           s3Client.putObject(putObj, filePath);
           System.out.println("Successfully placed " + objectKey +" into bucket
 "+bucketName);

       }

       catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
}
```

SDK for Python

Example

```
import boto3
import botocore
from botocore.exceptions import ClientError


def put_object(s3_client, bucket_name, key_name, object_bytes):
    """
    Upload data to a directory bucket.
    :param s3_client: The boto3 S3 client
    :param bucket_name: The bucket that will contain the object
    :param key_name: The key of the object to be uploaded
    :param object_bytes: The data to upload
    """
    try:
        response = s3_client.put_object(Bucket=bucket_name, Key=key_name,
                                        Body=object_bytes)
        print(f"Upload object '{key_name}' to bucket '{bucket_name}'.")
        return response
    except ClientError:
        print(f"Couldn't upload object '{key_name}' to bucket '{bucket_name}'.")
        raise

def main():
    # Share the client session with functions and objects to benefit from S3 Express
 One Zone auth key
    s3_client = boto3.client('s3')
    # Directory bucket name must end with --azid--x-s3
    resp = put_object(s3_client, 'doc-bucket-example--use1-az5--x-s3', 'sample.txt',
 b'Hello, World!')
    print(resp)

if __name__ == "__main__":
    main()
```

使用 AWS CLI

以下 `put-object` 示例命令显示如何使用 AWS CLI 从 Amazon S3 上传对象。要运行此命令，请将 *`user input placeholders`* 替换为您自己的信息。

```
aws s3api put-object --bucket bucket-base-name--azid--x-s3 --key sampleinut/file001.bin
 --body bucket-seed/file001.bin
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 put-object。

对目录桶使用分段上传

您可以使用分段上传过程将单个对象作为一组分段上传。每个分段都是对象数据的连续部分。您可以独立上传以及按任意顺序上传这些对象分段。如果任意分段传输失败，可以重新传输该分段且不会影响其他分段。上传完所有的对象分段后，Amazon S3 将汇集这些分段并创建对象。一般而言，如果您的对象大小达到了 100 MB，您应该考虑使用分段上传，而不是在单个操作中上传对象。

使用分段上传可提供以下优势：

- 提高吞吐量 – 您可以并行上传分段以提高吞吐量。
- 从任何网络问题中快速恢复 – 较小的分段大小可以将由于网络错误而重启失败的上传所产生的影响降至最低。
- 暂停和恢复对象上传 – 您可以在一段时间内逐步上传对象分段。启动分段上传后，没有过期日期。您必须显式完成或中止分段上传。
- 在您知道对象的最终大小前开始上传 – 您可以在创建对象时将其上传。

我们建议您按以下方式使用分段上传：

- 如果您在稳定的高带宽网络上传大型对象，请通过并行分段上传对象实现多线程性能，从而使用分段上传来充分利用您的可用带宽。
- 如果您在不稳定的网络中上传对象，请使用分段上传来提高应对网络错误的弹性，避免重启上传。在使用分段上传时，您只需重新尝试上传在上传期间中断的分段即可。而无需从头重新开始上传对象。

当您使用分段上传将对象上传到目录桶中的 Amazon S3 Express One Zone 存储类时，分段上传过程与使用分段上传将对象上传到通用桶的过程类似。但是，二者之间仍存在一些显著区别。

有关使用分段上传将对象上传到 S3 Express One Zone 的更多信息，请参阅以下主题。

主题

分段上传过程

分段上传的过程包括三个步骤：

- 启动上传。
- 上传对象分段。
- 上传完所有分段后，即完成分段上传。

在收到完成分段上传请求后，Amazon S3 会利用上传的分段构造对象，然后您可以像在您的桶中访问任何其它对象一样访问该对象。

分段上传开始

当您发送请求以开始分段上传时，Amazon S3 将返回具有上传 ID 的响应，此 ID 是分段上传的唯一标识符。无论您何时上传分段、列出分段、完成上传或中止上传，您都必须包括此上传 ID。

分段上传

上传分段时，除了指定上传 ID，还必须指定分段编号。在 S3 Express One Zone 中使用分段上传时，分段编号必须是连续的编号。如果您尝试使用非连续分段编号完成分段上传请求，则会生成 HTTP `400 Bad Request`（分段顺序无效）错误。

分段编号在您正在上传的对象中唯一地识别分段及其位置。如果您使用之前上传的分段的同一分段编号上传新分段，则之前上传的分段将被覆盖。

无论您何时上传分段，Amazon S3 都将在其响应中返回实体标签 (ETag) 标头。对于每个分段上传，您必须记录分段编号和 ETag 值。所有对象分段上传的 ETag 值将保持不变，但将为每个分段分配不同的分段编号。您必须在随后的请求中包括这些值以完成分段上传。

Amazon S3 会自动加密上传到 S3 存储桶的所有新对象。在进行分段上传时，如果您未在请求中指定加密信息，则已上传分段的加密设置将设为目标存储桶的默认加密配置。Amazon S3 存储桶的默认加密配置始终处于启用状态，并至少设置为使用 Amazon S3 托管密钥的服务器端加密（SSE-S3）。对于目录桶，仅支持 SSE-S3。有关更多信息，请参阅 具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）。

分段上传完成

完成分段上传时，Amazon S3 会按分段编号的升序将各个分段连接起来，从而创建对象。成功完成请求后，分段将不再存在。

完成分段上传请求必须包括上传 ID 以及分段编号及其相应的 ETag 值的列表。Amazon S3 响应包括可唯一地识别组合对象数据的 ETag。此 ETag 并非对象数据的 MD5 哈希。

分段上传列表

您可以列出特定分段上传或所有正在进行的分段上传的分段。列出分段操作将返回您已为特定分段上传而上传的分段信息。对于每个列出分段请求，Amazon S3 将返回有关特定分段上传的分段信息，最多为 1000 个分段。如果分段上传中的分段超过 1000 个，则必须使用分页来检索所有分段。

返回的分段列表不包括未完成上传的分段。使用列出分段上传操作，您可以获得正在进行的分段上传的列表。

正在进行的分段上传是已开始但还未完成或中止的上传。每个请求将返回最多 1000 个分段上传。如果正在进行的分段上传超过 1000 个，您必须发送其他请求才能检索剩余的分段上传。仅使用返回的列表进行验证。发送完成分段上传 请求时，请勿使用此列表的结果。相反，当上传分段和 Amazon S3 返回的相应 ETag 值时，请保留您自己的指定分段编号的列表。

有关分段上传列表的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 ListParts。

使用分段上传操作的校验和

上传对象时，可指定校验和算法来检查对象的完整性。目录存储桶不支持 MD5。您可以指定以下安全哈希算法（SHA）或循环冗余校验（CRC）数据完整性检查算法之一：

- CRC32
- CRC32C
- SHA-1
- SHA-256

您可以使用 Amazon S3 REST API 或 AWS SDK，通过 `GetObject` 或 `HeadObject` 来检索个别分段的校验和值。如果您想检索仍在进行的分段上传的各个分段的校验和值，可以使用 `ListParts`。

> **Important**
>
> 在使用前面的校验和算法时，分段编号必须是连续的编号。如果您尝试使用非连续分段编号完成分段上传请求，Amazon S3 会生成 `HTTP 400 Bad Request`（分段顺序无效）错误。

有关校验和如何处理分段对象的更多信息，请参阅检查对象完整性。

并发分段上传操作

在分布式开发环境中，您的应用程序可以同时在同一对象上启动多个更新。例如，您的应用程序可能会使用同一对象键启动多个分段上传。然后，对于其中每个上传，您的应用程序可以上传分段并将完成上传请求发送到 Amazon S3，以创建对象。对于 S3 Express One Zone，对象创建时间是分段上传的完成日期。

> **Important**
>
> 对于存储在目录桶中的对象，不支持版本控制。

分段上传和定价

开始分段上传后，Amazon S3 将保留所有分段，直到您完成或中止上传。在整个其生命周期内，您将支付有关此分段上传及其关联分段的所有存储、带宽和请求的费用。如果您中止分段上传，Amazon S3 将删除上传构件和已上传的任何分段，您将不再为它们支付费用。无论指定的存储类如何，删除未完成的分段上传均不收取提前删除费用。有关定价的更多信息，请参阅 Amazon S3 定价。

> **Important**
>
> 如果未成功发送完成分段上传请求，则不会汇集对象分段且不会创建对象。您需要为与上传的分段关联的所有存储付费。完成分段上传来创建对象，或者中止分段上传来移除任何已上传的分段非常重要。
>
> 在可删除目录桶之前，必须完成或中止所有正在进行的分段上传。目录桶不支持 S3 生命周期配置。如果需要，您可以列出正在进行的分段上传，然后中止上传并删除桶。

分段上传 API 操作和权限

要允许访问目录桶上的对象管理 API 操作，您需要在桶策略或基于 AWS Identity and Access Management（IAM）身份的策略中授予 `s3express:CreateSession` 权限。

您必须具有使用分段上传操作的所需权限。您可以使用桶策略或基于 IAM 身份的策略向 IAM 主体授予执行这些操作的权限。下表列出了各种分段上传操作所需的权限。

您可以通过 `Initiator` 元素识别分段上传的发起者。如果发起者是 AWS 账户，此元素将提供与 `Owner` 元素相同的信息。如果发起者是 IAM 用户，此元素将提供用户 ARN 和显示名称。

操作	所需的权限
创建分段上传	要创建分段上传，您必须得到可以对目录桶执行 `s3express:CreateSession` 操作的许可。
启动分段上传	要启动分段上传，您必须得到可以对目录桶执行 `s3express:CreateSession` 操作的许可。
上传分段	要上传分段，您必须得到可以对目录桶执行 `s3express:CreateSession` 操作的许可。 要使发起者能够上传分段，桶拥有者必须允许发起者对目录桶执行 `s3express:CreateSession` 操作。
上传分段（复制）	要上传分段，您必须得到可以对目录桶执行 `s3express:CreateSession` 操作的许可。 存储桶拥有者必须允许发起者对对象执行 `s3express:CreateSession` 操作，发起者才能上传该对象的分段。
完成分段上传	要完成分段上传，您必须得到可以对目录桶执行 `s3express:CreateSession` 操作的许可。 要使发起者能够完成分段上传，桶拥有者必须允许发起者对于对象执行 `s3express:CreateSession` 操作。
中止分段上传	要中止分段上传，您必须得到可以执行 `s3express:CreateSession` 操作的许可。

操作	所需的权限
	要使发起者能够中止分段上传，必须向发起者授予执行 `s3express:CreateSession` 操作的显式允许访问权限。
列出分段	要列出分段上传中的分段，您必须得到可以对目录桶执行 `s3express:CreateSession` 操作的许可。
列出正在进行的分段上传	要列出正在上传到桶的分段上传，您必须得到可以对桶执行 `s3:ListBucketMultipartUploads` 操作的许可。

分段上传的 API 操作支持

《Amazon Simple Storage Service API 参考》的下面几节描述了适用于分段上传的 Amazon S3 REST API 操作。

- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- AbortMultipartUpload
- ListParts
- ListMultipartUploads

示例

要使用分段上传将对象上传到目录桶中的 S3 Express One Zone，请参阅以下示例。

主题

- 创建分段上传
- 上传分段上传的分段
- 完成分段上传
- 中止分段上传
- 创建分段上传复制操作
- 列出正在进行的分段上传

- 列出分段上传的分段

创建分段上传

以下示例显示如何创建分段上传。

使用 AWS SDK

SDK for Java 2.x

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts
 *
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--use1-az4--x-s3'
 * @param key
 * @return
 */
 private static String createMultipartUpload(S3Client s3, String bucketName, String
 key) {

     CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
             .bucket(bucketName)
             .key(key)
             .build();

     String uploadId = null;

     try {
         CreateMultipartUploadResponse response =
 s3.createMultipartUpload(createMultipartUploadRequest);
         uploadId = response.uploadId();
     }
     catch (S3Exception e) {
         System.err.println(e.awsErrorDetails().errorMessage());
         System.exit(1);
     }
     return uploadId;
```

SDK for Python

Example

```
def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
   Create a multipart upload to a directory bucket

   :param s3_client: boto3 S3 client
   :param bucket_name: The destination bucket for the multipart upload
   :param key_name: The key name for the object to be uploaded
   :return: The UploadId for the multipart upload if created successfully, else None
   '''

   try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None
```

使用 AWS CLI

此示例说明如何使用 AWS CLI 创建到目录存储桶的分段上传。此命令启动到对象 *KEY_NAME* 的目录存储桶 *bucket-base-name*--*azid*--x-s3 的分段上传。要使用该命令，请将*#######* 替换为您自己的信息。

```
aws s3api create-multipart-upload --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

有关更多信息，请参阅 AWS Command Line Interface 中的 create-multipart-upload。

上传分段上传的分段

以下示例显示了如何上传分段上传的各分段。

使用 AWS SDK

SDK for Java 2.x

以下示例显示了如何使用 SDK for Java 2.x 将单个对象分段，然后将这些分段上传到目录存储桶。

Example

```
/**
 * This method creates part requests and uploads individual parts to S3 and then
 returns all the completed parts
 *
 * @param s3
 * @param bucketName
 * @param key
 * @param uploadId
 * @throws IOException
 */
 private static ListCompletedPartmultipartUpload(S3Client s3, String bucketName,
 String key, String uploadId, String filePath) throws IOException {

        int partNumber = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // read the local file, breakdown into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            int position = 0;
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
 buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();
```

```
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        }

        catch (IOException e) {
            throw e;
        }
        return completedParts;
    }
```

SDK for Python

以下示例显示了如何使用 SDK for Python 将单个对象分段，然后将这些分段上传到目录存储桶。

Example

```
def multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_size):
    '''
    Break up a file into multiple parts and upload those parts to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name for object to be uploaded and for the local file
 that's being uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart upload if all parts are uploaded
 successfully, else None
    '''

    part_list = []
    try:
        with open(key_name, 'rb') as file:
            part_counter = 1
            while True:
                file_part = file.read(part_size)
                if not len(file_part):
                    break
```

```
                upload_part = s3_client.upload_part(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = mpu_id,
                    Body = file_part,
                    PartNumber = part_counter
                )
                part_list.append({'PartNumber': part_counter, 'ETag':
upload_part['ETag']})
                part_counter += 1
   except ClientError as e:
       logging.error(e)
       return None
   return part_list
```

使用 AWS CLI

此示例显示了如何使用 AWS CLI 将单个对象分段，然后将这些分段上传到目录存储桶。要使用该命令，请将*#######* 替换为您自己的信息。

```
aws s3api upload-part --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME --part-number 1 --body LOCAL_FILE_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

有关更多信息，请参阅 AWS Command Line Interface 中的 upload-part。

完成分段上传

以下示例显示了如何完成分段上传。

使用 AWS SDK

SDK for Java 2.x

以下示例显示了如何使用 SDK for Java 2.x 完成分段上传。

Example

```
/**
 * This method completes the multipart upload request by collating all the upload
 parts
 * @param s3
```

```
* @param bucketName - for example, 'doc-example-bucket--usw2-az1--x-s3'
* @param key
* @param uploadId
* @param uploadParts
*/
private static void completeMultipartUpload(S3Client s3, String bucketName, String
key, String uploadId, ListCompletedPart uploadParts) {
       CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
               .parts(uploadParts)
               .build();

       CompleteMultipartUploadRequest completeMultipartUploadRequest =
               CompleteMultipartUploadRequest.builder()
                       .bucket(bucketName)
                       .key(key)
                       .uploadId(uploadId)
                       .multipartUpload(completedMultipartUpload)
                       .build();

       s3.completeMultipartUpload(completeMultipartUploadRequest);
   }

   public static void multipartUploadTest(S3Client s3, String bucketName, String
key, String localFilePath)  {
       System.out.println("Starting multipart upload for: " + key);
       try {
           String uploadId = createMultipartUpload(s3, bucketName, key);
           System.out.println(uploadId);
           ListCompletedPart parts = multipartUpload(s3, bucketName, key, uploadId,
localFilePath);
           completeMultipartUpload(s3, bucketName, key, uploadId, parts);
           System.out.println("Multipart upload completed for: " + key);
       }

       catch (Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

SDK for Python

以下示例显示了如何使用 SDK for Python 完成分段上传。

Example

```
def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: The destination bucket for the multipart upload
    :param key_name: The key name for the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: The list of uploaded part numbers with their associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'OBJECT_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_upload(s3_client, bucket_name, key_name, mpu_id,
 part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id,
 part_list):
                print (f'{key_name} successfully uploaded through a ultipart upload
 to {bucket_name}')
```

```
            else:
                print (f'Could not upload {key_name} hrough a multipart upload to
{bucket_name}')
```

使用 AWS CLI

此示例说明如何使用 AWS CLI 完成目录存储桶的分段上传。要使用该命令，请将####### 替换为您自己的信息。

```
aws s3api complete-multipart-upload --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
 --multipart-upload file://parts.json
```

此示例采用 JSON 结构，用于描述分段上传中应重新组合成完整文件的各个分段。在此示例中，file:// 前缀用于从名为 parts 的本地文件夹中的文件加载 JSON 结构。

parts.json：

```
parts.json
{
  "Parts": [
    {
      "ETag": "6b78c4a64dd641a58dac8d9258b88147",
      "PartNumber": 1
    }
  ]
}
```

有关更多信息，请参阅 AWS Command Line Interface 中的 complete-multipart-upload。

中止分段上传

以下示例显示了如何中止分段上传。

使用 AWS SDK

SDK for Java 2.x

以下示例显示了如何使用 SDK for Java 2.x 中止分段上传。

Example

```
public static void abortMultiPartUploads( S3Client s3, String bucketName ) {

        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();

            AbortMultipartUploadRequest abortMultipartUploadRequest;
            for (MultipartUpload upload: uploads) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(upload.key())
                        .uploadId(upload.uploadId())
                        .build();

                s3.abortMultipartUpload(abortMultipartUploadRequest);
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

以下示例显示了如何使用 SDK for Python 中止分段上传。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
def abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
    '''
    Aborts a partial multipart upload in a directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket where the multipart upload was initiated - for
 example, 'doc-example-bucket--usw2-az1--x-s3'
    :param key_name: Name of the object for which the multipart upload needs to be
 aborted
    :param upload_id: Multipart upload ID for the multipart upload to be aborted
    :return: True if the multipart upload was successfully aborted, False if not
    '''
    try:
        s3_client.abort_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = upload_id
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
        upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    if abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
        print (f'Multipart upload for object {key_name} in {bucket_name} bucket has
 been aborted')
    else:
        print (f'Unable to abort multipart upload for object {key_name} in
 {bucket_name} bucket')
```

使用 AWS CLI

以下示例显示了如何使用 AWS CLI 中止分段上传。要使用该命令，请将####### 替换为您自己的信息。

```
aws s3api abort-multipart-upload --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
 --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
MAQAAAAB0OxUFeA7LTbWWFS8WYwhrxDxTIDN-pdEEq_agIHqsbg"
```

有关更多信息，请参阅 AWS Command Line Interface 中的 abort-multipart-upload。

创建分段上传复制操作

以下示例显示了如何使用分段上传将对象从一个存储桶复制到另一个存储桶。

使用 AWS SDK

SDK for Java 2.x

以下示例显示了如何使用 SDK for Java 2.x，通过分段上传以编程方式将对象从一个存储桶复制到另一个存储桶。

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts.
 *
 * @param s3
 * @param bucketName
 * @param key
 * @return
 */
 private static String createMultipartUpload(S3Client s3, String bucketName, String
 key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();
        String uploadId = null;
        try {
            CreateMultipartUploadResponse response =
 s3.createMultipartUpload(createMultipartUploadRequest);
            uploadId = response.uploadId();
        } catch (S3Exception e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return uploadId;
 }

 /**
  * Creates copy parts based on source object size and copies over individual parts
  *
  * @param s3
  * @param sourceBucket
  * @param sourceKey
  * @param destnBucket
  * @param destnKey
  * @param uploadId
  * @return
  * @throws IOException
  */
   public static ListCompletedPart multipartUploadCopy(S3Client s3, String
sourceBucket, String sourceKey, String destnBucket, String destnKey, String
uploadId) throws IOException {

        // Get the object size to track the end of the copy operation.
        HeadObjectRequest headObjectRequest = HeadObjectRequest
                .builder()
                .bucket(sourceBucket)
                .key(sourceKey)
                .build();
        HeadObjectResponse response = s3.headObject(headObjectRequest);
        Long objectSize = response.contentLength();

        System.out.println("Source Object size: " + objectSize);

        // Copy the object using 20 MB parts.
        long partSize = 20 * 1024 * 1024;
        long bytePosition = 0;
        int partNum = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        while (bytePosition < objectSize) {
            // The last part might be smaller than partSize, so check to make sure
            // that lastByte isn't beyond the end of the object.
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);
```

```
            System.out.println("part no: " + partNum + ", bytePosition: " +
bytePosition + ", lastByte: " + lastByte);

            // Copy this part.
            UploadPartCopyRequest req = UploadPartCopyRequest.builder()
                    .uploadId(uploadId)
                    .sourceBucket(sourceBucket)
                    .sourceKey(sourceKey)
                    .destinationBucket(destnBucket)
                    .destinationKey(destnKey)
                    .copySourceRange("bytes="+bytePosition+"-"+lastByte)
                    .partNumber(partNum)
                    .build();
            UploadPartCopyResponse res = s3.uploadPartCopy(req);
            CompletedPart part = CompletedPart.builder()
                    .partNumber(partNum)
                    .eTag(res.copyPartResult().eTag())
                    .build();
            completedParts.add(part);
            partNum++;
            bytePosition += partSize;
        }
        return completedParts;
    }


    public static void multipartCopyUploadTest(S3Client s3, String srcBucket, String
srcKey, String destnBucket, String destnKey)  {
        System.out.println("Starting multipart copy for: " + srcKey);
        try {
            String uploadId = createMultipartUpload(s3, destnBucket, destnKey);
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUploadCopy(s3, srcBucket,
srcKey,destnBucket,  destnKey, uploadId);
            completeMultipartUpload(s3, destnBucket, destnKey, uploadId, parts);
            System.out.println("Multipart copy completed for: " + srcKey);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python

以下示例显示了如何使用 SDK for Python，通过分段上传以编程方式将对象从一个存储桶复制到另一个存储桶。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def head_object(s3_client, bucket_name, key_name):
    '''
    Returns metadata for an object in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains the object to query for metadata
    :param key_name: Key name to query for metadata
    :return: Metadata for the specified object if successful, else None
    '''

    try:
        response = s3_client.head_object(
            Bucket = bucket_name,
            Key = key_name
        )
        return response
    except ClientError as e:
        logging.error(e)
        return None

def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
    Create a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :return: UploadId for the multipart upload if created successfully, else None
    '''

    try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
```

```
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None

def multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size):
    '''
    Copy an object in a directory bucket to another bucket in multiple parts of a
 specified size

    :param s3_client: boto3 S3 client
    :param source_bucket_name: Bucket where the source object exists
    :param key_name: Key name of the object to be copied
    :param target_bucket_name: Destination bucket for copied object
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart copy if all parts are copied successfully,
 else None
    '''

    part_list = []
    copy_source = {
        'Bucket': source_bucket_name,
        'Key': key_name
    }
    try:
        part_counter = 1
        object_size = head_object(s3_client, source_bucket_name, key_name)
        if object_size is not None:
            object_size = object_size['ContentLength']
        while (part_counter - 1) * part_size <object_size:
            bytes_start = (part_counter - 1) * part_size
            bytes_end = (part_counter * part_size) - 1
            upload_copy_part = s3_client.upload_part_copy (
                Bucket = target_bucket_name,
                CopySource = copy_source,
                CopySourceRange = f'bytes={bytes_start}-{bytes_end}',
                Key = key_name,
                PartNumber = part_counter,
                UploadId = mpu_id
            )
```

```
            part_list.append({'PartNumber': part_counter, 'ETag':
 upload_copy_part['CopyPartResult']['ETag']})
            part_counter += 1
    except ClientError as e:
        logging.error(e)
        return None
    return part_list

def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: List of uploaded part numbers with associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    source_bucket_name = 'SOURCE_BUCKET_NAME'
    target_bucket_name = 'TARGET_BUCKET_NAME'
    key_name = 'KEY_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, target_bucket_name, key_name)
    if mpu_id is not None:
```

```
        part_list = multipart_copy_upload(s3_client, source_bucket_name, key_name,
target_bucket_name, mpu_id, part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, target_bucket_name, key_name,
mpu_id, part_list):
                print (f'{key_name} successfully copied through multipart copy from
{source_bucket_name} to {target_bucket_name}')
            else:
                print (f'Could not copy {key_name} through multipart copy from
{source_bucket_name} to {target_bucket_name}')
```

使用 AWS CLI

以下示例显示了如何使用 AWS CLI，通过分段上传以编程方式将对象从一个存储桶复制到目录存储桶。要使用该命令，请将*#######* 替换为您自己的信息。

```
aws s3api upload-part-copy --bucket bucket-base-name--azid--x-s3 --key TARGET_KEY_NAME
 --copy-source SOURCE_BUCKET_NAME/SOURCE_KEY_NAME --part-number 1 --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

有关更多信息，请参阅 AWS Command Line Interface 中的 upload-part-copy。

列出正在进行的分段上传

要列出正在进行的到目录存储桶的分段上传，可以使用 AWS SDK 或 AWS CLI。

使用 AWS SDK

SDK for Java 2.x

以下示例显示了如何使用 SDK for Java 2.x 列出正在进行（未完成）的分段上传。

Example

```
public static void listMultiPartUploads( S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

            ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
```

```
            List MultipartUpload uploads = response.uploads();
            for (MultipartUpload upload: uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key() +
 "\", id = " + upload.uploadId());
            }
      }
      catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
      }
  }
```

SDK for Python

以下示例显示了如何使用 SDK for Python 列出正在进行（未完成）的分段上传。

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_multipart_uploads(s3_client, bucket_name):
    '''
    List any incomplete multipart uploads in a directory bucket in e specified gion

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to check for incomplete multipart uploads
    :return: List of incomplete multipart uploads if there are any, None if not
    '''

    try:
        response = s3_client.list_multipart_uploads(Bucket = bucket_name)
        if 'Uploads' in response.keys():
            return response['Uploads']
        else:
            return None
    except ClientError as e:
        logging.error(e)

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
```

```
    s3_client = boto3.client('s3', region_name = region)
    multipart_uploads = list_multipart_uploads(s3_client, bucket_name)
    if multipart_uploads is not None:
        print (f'There are {len(multipart_uploads)} ncomplete multipart uploads for
 {bucket_name}')
    else:
        print (f'There are no incomplete multipart uploads for {bucket_name}')
```

使用 AWS CLI

以下示例显示了如何使用 AWS CLI 列出正在进行（未完成）的分段上传。要使用该命令，请将*#####
##* 替换为您自己的信息。

```
aws s3api list-multipart-uploads --bucket bucket-base-name--azid--x-s3
```

有关更多信息，请参阅 AWS Command Line Interface 中的 list-multipart-uploads。

列出分段上传的分段

以下示例显示了如何列出到目录存储桶的分段上传的各个分段。

使用 AWS SDK

SDK for Java 2.x

以下示例显示了如何使用 SDK for Java 2.x 列出到目录存储桶的分段上传的各个分段。

```
public static void listMultiPartUploadsParts( S3Client s3, String bucketName, String
 objKey, String uploadID) {

        try {
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                .bucket(bucketName)
                .uploadId(uploadID)
                .key(objKey)
                .build();

            ListPartsResponse response = s3.listParts(listPartsRequest);
            ListPart parts = response.parts();
            for (Part part: parts) {
                System.out.println("Upload in progress: Part number = \"" +
 part.partNumber() + "\", etag = " + part.eTag());
```

```
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }


    }
```

SDK for Python

以下示例显示了如何使用 SDK for Python 列出到目录存储桶的分段上传的各个分段。

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_parts(s3_client, bucket_name, key_name, upload_id):
    '''
    Lists the parts that have been uploaded for a specific multipart upload to a
 directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that multipart uploads parts have been uploaded to
    :param key_name: Name of the object that has parts uploaded
    :param upload_id: Multipart upload ID that the parts are associated with
    :return: List of parts associated with the specified multipart upload, None if
 there are no parts
    '''
    parts_list = []
    next_part_marker = ''
    continuation_flag = True
    try:
        while continuation_flag:
            if next_part_marker == '':
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id
                )
            else:
```

```
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id,
                    NextPartMarker = next_part_marker
                )
            if 'Parts' in response:
                for part in response['Parts']:
                    parts_list.append(part)
                if response['IsTruncated']:
                    next_part_marker = response['NextPartNumberMarker']
                else:
                    continuation_flag = False
            else:
                continuation_flag = False
        return parts_list
    except ClientError as e:
        logging.error(e)
        return None

if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
    upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    parts_list = list_parts(s3_client, bucket_name, key_name, upload_id)
    if parts_list is not None:
        print (f'{key_name} has {len(parts_list)} parts uploaded to {bucket_name}')
    else:
        print (f'There are no multipart uploads with that upload ID for
 {bucket_name} bucket')
```

使用 AWS CLI

以下示例显示了如何使用 AWS CLI 列出到目录存储桶的分段上传的各个分段。要使用该命令，请将*## ##### * 替换为您自己的信息。

```
aws s3api list-parts --bucket bucket-base-name--azid--x-s3 --key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

有关更多信息，请参阅 AWS Command Line Interface 中的 list-parts。

将对象复制到目录存储桶

复制操作将创建已存储在 Amazon S3 中的对象的副本。您可以在目录存储桶和通用存储桶之间复制对象。您还可以在一个存储桶内复制对象，也可以在同一类型的存储桶之间复制对象，例如，从目录存储桶复制到目录存储桶。

在单个原子操作中，您可以创建最大 5GB 的对象副本。但是，要复制大于 5GB 的对象，您必须使用分段上传 API 操作。有关更多信息，请参阅 对目录桶使用分段上传。

权限

要复制对象，您必须拥有以下权限：

- 要将对象从一个目录存储桶复制到另一个目录存储桶，您必须拥有 `s3express:CreateSession` 权限。
- 要将对象从目录存储桶复制到通用存储桶，您必须拥有将对象副本写入目标存储桶的 `s3express:CreateSession` 权限和 `s3:PutObject` 权限。
- 要将对象从通用桶复制到目录桶，您必须拥有读取正在复制的源对象的 `s3express:CreateSession` 权限和 `s3:GetObject` 权限。

  有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CopyObject。

加密

Amazon S3 会自动加密上传到 S3 存储桶的所有新对象。S3 存储桶的默认加密配置始终处于启用状态，并至少设置为具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）。

对于目录桶，仅支持 SSE-S3。对于通用桶，您可以使用 SSE-S3（默认）、采用 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、采用 AWS KMS 密钥的双层服务器端加密（DSSE-KMS），或采用客户提供密钥的服务器端加密（SSE-C）。

如果您发出复制请求将目录桶上的 SSE-C、SSE-KMS 或 DSSE-KMS 参数设置为源或目标，则响应将返回错误。

标签

目录存储桶不支持标签。如果您将带有标签的对象从通用桶复制到目录桶，您会收到 HTTP `501 (Not Implemented)` 响应。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CopyObject。

ETag

S3 Express One Zone 的实体标签（ETag）是随机的字母数字字符串，而不是 MD5 校验和。为协助确保对象完整性，请使用额外的校验和。

其他校验和

S3 Express One Zone 可让您选择用于在上传或下载过程中验证数据的校验和算法。您可以选择以下安全哈希算法（SHA）或循环冗余校验（CRC）数据完整性检查算法之一：CRC32、CRC32C、SHA-1 和 SHA-256。S3 Express One Zone 存储类不支持基于 MD5 的校验和。

有关更多信息，请参阅 其他 S3 校验和最佳实践。

支持的特征

有关 S3 Express One Zone 支持哪些 Amazon S3 特征的更多信息，请参阅S3 Express One Zone 有哪些不同？。

使用 S3 控制台（复制到目录存储桶）

将对象从通用桶或目录桶复制到目录桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择要从中复制对象的桶：
   - 要从通用桶进行复制，请选择通用桶选项卡。
   - 要从目录桶进行复制，请选择目录桶选项卡。
4. 选择包含要复制的对象的通用桶或目录桶。
5. 请选择对象选项卡。在对象页面上，选中要复制的对象名称左侧的复选框。
6. 在 Actions (操作) 菜单中，选择 Copy (复制)。

   将出现复制页面。
7. 在目标下，为您的目标类型选择目录桶。要指定目标路径，请选择浏览 S3，导航到目标，然后选中目标左侧的选项按钮。选择右下角的选择目标。

   或者，输入目标路径。
8. 在校验和下，选择是要使用现有校验和函数复制对象，还是用新校验和函数替换现有校验和函数。上传对象时，您可以选择指定用于验证数据完整性的校验和算法。复制对象时，您可以选择新函数。如果您最初没有指定额外的校验和，则可以使用校验和部分添加一个校验和。

> **Note**
>
> 即使您选择使用相同的校验和函数，但如果对象大小超过 16MB，校验和值也可能会发生变化。由于分段上传的校验和计算方式的原因，校验和值可能会发生变化。有关在复制对象时校验和可能会如何变化的信息，请参阅使用分段级别墅校验和进行分段上传。

要更改校验和函数，请选择用新的校验和函数替换。从下拉列表中选择新的校验和函数。复制对象时，将使用指定的算法计算和存储新校验和。

9. 选择右下角的复制。Amazon S3 会将对象复制到目标。

使用 S3 控制台（复制到通用存储桶）

将对象从目录存储桶复制到通用存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择目录桶选项卡。
4. 选择包含您要复制的对象的目录桶。
5. 请选择对象选项卡。在对象页面上，选中要复制的对象名称左侧的复选框。
6. 在 Actions (操作) 菜单中，选择 Copy (复制)。
7. 在目标下，为您的目标类型选择通用桶。要指定目标路径，请选择浏览 S3，导航到目标，然后选中目标左侧的选项按钮。选择右下角的选择目标。

   或者，输入目标路径。
8. 在校验和下，选择是要使用现有校验和函数复制对象，还是用新校验和函数替换现有校验和函数。上传对象时，您可以选择指定用于验证数据完整性的校验和算法。复制对象时，您可以选择新函数。如果您最初没有指定额外的校验和，则可以使用校验和部分添加一个校验和。

> **Note**
>
> 即使您选择使用相同的校验和函数，但如果对象大小超过 16MB，校验和值也可能会发生变化。由于分段上传的校验和计算方式的原因，校验和值可能会发生变化。有关在复制对象时校验和可能会如何变化的信息，请参阅使用分段级别墅校验和进行分段上传。

要更改校验和函数，请选择用新的校验和函数替换。从下拉列表中选择新的校验和函数。复制对象时，将使用指定的算法计算和存储新校验和。

9. 选择右下角的复制。Amazon S3 会将对象复制到目标。

使用 AWS SDK

SDK for Java 2.x

Example

```
public static void copyBucketObject (S3Client s3, String sourceBucket, String
objectKey, String targetBucket) {
     CopyObjectRequest copyReq = CopyObjectRequest.builder()
         .sourceBucket(sourceBucket)
         .sourceKey(objectKey)
         .destinationBucket(targetBucket)
         .destinationKey(objectKey)
         .build();
      String temp = "";

      try {
          CopyObjectResponse copyRes = s3.copyObject(copyReq);
          System.out.println("Successfully copied " + objectKey +" from bucket " +
sourceBucket +" into bucket "+targetBucket);
      }

      catch (S3Exception e) {
          System.err.println(e.awsErrorDetails().errorMessage());
          System.exit(1);
      }
}
```

使用 AWS CLI

以下 copy-object 示例命令显示了如何使用 AWS CLI 将对象从一个存储桶复制到另一个存储库。您可以在存储桶类型之间复制对象。要运行此命令，请将用户输入占位符替换为您自己的信息。

```
aws s3api copy-object --copy-source bucket SOURCE_BUCKET/SOURCE_KEY_NAME --
key TARGET_KEY_NAME --bucket TARGET_BUCKET_NAME
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 copy-object。

删除目录存储桶中的对象

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）或 AWS SDK 从 Amazon S3 目录桶中删除对象。有关更多信息，请参阅 目录桶和 什么是 S3 Express One Zone？。

Warning

- 删除对象的操作无法撤销。
- 此操作将删除所有指定的对象。删除文件夹时，请等待删除操作完成，然后再将新对象添加到文件夹。否则，新对象也可能会被删除。

Note

当以编程方式从目录桶中删除多个对象时，请注意以下事项：

- DeleteObjects 请求中的对象键必须至少包含一个非空格字符。不支持完全由空格字符组成的字符串。
- DeleteObjects 请求中的对象键不能包含 Unicode 控制字符，但换行符（\n）、制表符（\t）和回车符（\r）除外。

使用 S3 控制台

删除对象

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 在左侧导航窗格中，选择存储桶。
3. 选择目录桶选项卡。
4. 选择包含您要删除的对象的目录桶。
5. 请选择对象选项卡。在对象列表中，选中要删除的一个或多个对象左侧的复选框。
6. 选择 删除。
7. 在删除对象页面上，在文本框中输入 **permanently delete**。
8. 选择删除对象。

使用 AWS SDK

SDK for Java 2.x

Example

下面的示例通过使用 AWS SDK for Java 2.x 删除目录桶中的对象。

```
static void deleteObject(S3Client s3Client, String bucketName, String objectKey) {



        try {

            DeleteObjectRequest del = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            s3Client.deleteObject(del);

            System.out.println("Object " + objectKey + " has been deleted");


        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

    }
```

SDK for Python

Example

下面的示例通过使用 AWS SDK for Python (Boto3) 删除目录桶中的对象。

```
import logging
import boto3
from botocore.exceptions import ClientError

def delete_objects(s3_client, bucket_name, objects):
    '''
    Delete a list of objects in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains objects to be deleted; for example,
 'doc-example-bucket--usw2-az1--x-s3'
    :param objects: List of dictionaries that specify the key names to delete
    :return: Response output, else False
    '''

    try:
        response = s3_client.delete_objects(
            Bucket = bucket_name,
            Delete = {
                'Objects': objects
            }
        )
        return response
    except ClientError as e:
        logging.error(e)
        return False


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    objects = [
        {
            'Key': '0.txt'
        },
        {
            'Key': '1.txt'
```

```
        },
        {
            'Key': '2.txt'
        },
        {
            'Key': '3.txt'
        },
        {
            'Key': '4.txt'
        }
    ]

    s3_client = boto3.client('s3', region_name = region)
    results = delete_objects(s3_client, bucket_name, objects)
    if results is not None:
        if 'Deleted' in results:
            print (f'Deleted {len(results["Deleted"])} objects from {bucket_name}')
        if 'Errors' in results:
            print (f'Failed to delete {len(results["Errors"])} objects from
{bucket_name}')
```

使用 AWS CLI

以下 `delete-object` 示例命令显示如何使用 AWS CLI 从目录存储桶中删除对象。要运行此命令，请将 *`user input placeholders`* 替换为您自己的信息。

```
aws s3api delete-object --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 delete-object。

下载目录桶中的对象

下面的代码示例显示了如何使用 `GetObject` API 操作从 Amazon S3 目录桶中的对象读取（下载）数据。

使用 AWS SDK

SDK for Java 2.x

Example

下面的代码示例显示了如何使用 AWS SDK for Java 2.x 从目录桶中的对象读取数据。

```
public static void getObject(S3Client s3Client, String bucketName, String objectKey)
 {
     try {
         GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(objectKey)
            .bucket(bucketName)
            .build();

         ResponseBytes GetObjectResponse objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
         byte[] data = objectBytes.asByteArray();

         //Print object contents to console
         String s = new String(data, StandardCharsets.UTF_8);
         System.out.println(s);
    }

    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python

Example

下面的代码示例显示了如何使用 AWS SDK for Python (Boto3) 从目录桶中的对象读取数据。

```
import boto3
from botocore.exceptions import ClientError
from botocore.response import StreamingBody

def get_object(s3_client: boto3.client, bucket_name: str, key_name: str) ->
 StreamingBody:
    """
    Gets the object.
    :param s3_client:
    :param bucket_name: The bucket that contains the object.
    :param key_name: The key of the object to be downloaded.
    :return: The object data in bytes.
    """
```

```
    try:
        response = s3_client.get_object(Bucket=bucket_name, Key=key_name)
        body = response['Body'].read()
        print(f"Got object '{key_name}' from bucket '{bucket_name}'.")
    except ClientError:
        print(f"Couldn't get object '{key_name}' from bucket '{bucket_name}'.")
        raise
    else:
        return body

def main():
    s3_client = boto3.client('s3')
    resp = get_object(s3_client, 'doc-example-bucket--use1-az4--x-s3', 'sample.txt')
    print(resp)

if __name__ == "__main__":
     main()
```

使用 AWS CLI

以下 `get-object` 示例将向您展示如何使用 AWS CLI 从 Amazon S3 下载对象。此命令从目录存储桶 *bucket-base-name--azid--x-s3* 获取对象 *KEY_NAME*。该对象将下载到名为 *LOCAL_FILE_NAME* 的文件中。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3api get-object --bucket bucket-base-name--azid--x-s3 --
key KEY_NAME LOCAL_FILE_NAME
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 get-object。

将 HeadObject 与目录桶结合使用

以下 AWS SDK 和 AWS CLI 示例显示如何使用 `HeadObject` API 操作从 Amazon S3 目录存储桶中的对象检索元数据，而不返回对象本身。

使用 AWS SDK

SDK for Java 2.x

Example

```
public static void headObject(S3Client s3Client, String bucketName, String
 objectKey) {
    try {
        HeadObjectRequest headObjectRequest = HeadObjectRequest
                .builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();
        HeadObjectResponse response = s3Client.headObject(headObjectRequest);
        System.out.format("Amazon S3 object: \"%s\" found in bucket: \"%s\" with
 ETag: \"%s\"", objectKey, bucketName, response.eTag());
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
```

使用 AWS CLI

以下 head-object 示例命令显示您如何使用 AWS CLI 从对象检索元数据。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3api head-object --bucket bucket-base-name--azid--x-s3 --key KEY_NAME
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 head-object。

S3 Express One Zone 的安全性

AWS 十分重视云安全性。为了满足对安全性最敏感的组织的需求，我们打造了具有超高安全性的数据中心和网络架构。作为 AWS 客户，您也将从这些数据中心和网络架构受益。安全性是 AWS 和您的共同责任。责任共担模式将其描述为云本身的安全性和云中的安全性：

- 云的安全性 – AWS 负责保护在 AWS Cloud 中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为 AWS Compliance Programs的一部分，第三方审计人员将定期测试和验证我们安全措施的有效性。

  要了解适用于 Amazon S3 Express One Zone 的合规性计划，请参阅AWS 服务 in Scope by Compliance Program。
- 云中的安全性 – 您的责任由您使用的 AWS 服务 决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将协助您了解在使用 S3 Express One Zone 时如何应用责任共担模式。以下主题说明如何配置 S3 Express One Zone 以实现您的安全性和合规性目标。您还将了解在使用 S3 Express One Zone 时，如何使用其他协助您监控和保护资源的 AWS 服务。

主题

数据保护和加密

有关 S3 Express One Zone 如何加密和保护数据的更多信息，请参阅以下主题。

主题

具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）

默认情况下，存储在目录存储桶中的所有对象，都通过具有 Amazon S3 托管加密密钥的服务器端加密（SSE-S3）进行加密。不允许将未加密的内容上传到目录存储桶。有关更多信息，请参阅 使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）和 利用加密来保护数据。

目录存储桶不支持具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、具有 AWS Key Management Service（AWS KMS）密钥的双层服务器端加密（DSSE-KMS）或者具有客户提供加密密钥的服务器端加密（SSE-C）。

传输中加密

只能通过 HTTPS（TLS）访问 S3 Express One Zone。

S3 Express One Zone 使用区域和可用区 API 端点。根据您使用的 Amazon S3 API 操作，需要区域端点或可用区端点。您可以通过网关虚拟私有云（VPC）端点访问区域端点和可用区端点。使用网关端点不会发生任何额外费用。要了解有关区域和可用区 API 端点的更多信息，请参阅S3 Express One Zone 联网。

其他校验和

S3 Express One Zone 可让您选择用于在上传或下载过程中验证数据的校验和算法。您可以选择以下安全哈希算法（SHA）或循环冗余校验（CRC）数据完整性检查算法之一：CRC32、CRC32C、SHA-1 和 SHA-256。S3 Express One Zone 存储类不支持基于 MD5 的校验和。

有关更多信息，请参阅其他 S3 校验和最佳实践。

数据删除

您可以使用 Amazon S3 控制台、AWS SDK、AWS Command Line Interface（AWS CLI）或 Amazon S3 REST API，直接从 S3 Express One Zone 删除一个或多个对象。由于目录存储桶中的所有对象都会产生存储费用，因此建议您删除不再需要的对象。

删除存储在目录存储桶中的对象也会递归地删除所有父目录，前提是在要删除的对象之外，这些父目录不包含其他任何对象。

> **Note**
>
> S3 Express One Zone 不支持多重身份验证（MFA）删除和 S3 版本控制。

适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）

AWS Identity and Access Management（IAM）是一项 AWS 服务，有助于管理员安全地控制对 AWS 资源的访问。IAM 管理员可控制哪些用户能够通过身份验证（登录）和获得授权（拥有权限），使用 S3 Express One Zone 中的 Amazon S3 资源。使用 IAM 不会产生额外的费用。

默认情况下，用户没有目录存储桶和 S3 Express One Zone 操作的权限。要授予对目录存储桶的访问权限，您可以使用 IAM 创建用户、组或角色，然后将权限附加到这些身份。有关 IAM 的更多信息，请参阅《IAM 用户指南》中的安全最佳实践。

要提供访问权限，您可以通过以下方法为用户、组或角色添加权限：

- AWS IAM Identity Center 中的用户和组 – 创建权限集。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。
- 通过身份提供商在 IAM 中管理的用户 – 创建身份联合验证角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。
- IAM 角色和用户 – 创建您的用户可以代入的角色。请按照《IAM 用户指南》的创建向 IAM 用户委派权限的角色中的说明进行操作。

默认情况下，目录存储桶为私有，只有被明确授予访问权限的用户才可以访问。目录存储桶的访问控制边界仅在存储桶级设置。与之相对，通用存储桶的访问控制边界可以在存储桶、前缀或对象标签级设置。这种不同意味着，对于 S3 Express One Zone 访问权限，目录存储桶是唯一可以包括在存储桶策略或 IAM 身份策略中的资源。

使用 S3 Express One Zone 时，除 IAM 授权之外，您还可以通过由 `CreateSession` API 操作处理的基于会话的新机制，对请求进行身份验证和授权。您可以使用 `CreateSession` 来请求临时凭证，以便提供对存储桶的低延迟访问。这些临时凭证的作用范围限制为特定的目录存储桶。

要使用 `CreateSession`，我们建议使用最新版本的 AWS SDK 或者使用 AWS Command Line Interface（AWS CLI）。支持的 AWS SDK 以及 AWS CLI 代表您处理会话的建立、刷新和终止。

您可以使用会话令牌以及仅限可用区（对象级）的操作（`CopyObject` 和 `HeadBucket` 除外），在一个会话中的多个请求上分配与授权关联的延迟。对于区域端点 API 操作（存储桶级操作），您可以使用 IAM 授权，这不涉及对会话的管理。有关更多信息，请参阅 适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）和 `CreateSession` 授权。

有关为 S3 Express One Zone 设置 IAM 的更多信息，请参阅以下主题。

主题

- 主体
- 资源
- 适用于 S3 Express One Zone 的操作
- 适用于 S3 Express One Zone 的条件键
- 如何对 API 操作进行授权和身份验证

主体

创建基于资源的策略以授予对存储桶的访问权限时，您必须使用 `Principal` 元素来指定可请求对该资源执行某个操作或运算的人员或应用程序。对于目录存储桶策略，您可以使用以下主体：

- AWS 账户
- IAM 用户
- IAM 角色
- 联合用户

有关更多信息，请参阅 IAM 用户指南中的 Principal。

资源

目录存储桶的 Amazon 资源名称（ARN）包含 s3express 命名空间、AWS 区域、AWS 账户 ID 和目录存储桶名称（包括可用区 ID）。要访问目录存储桶并对其执行操作，您必须使用以下 ARN 格式：

```
arn:aws:s3express:region:account-id:bucket/base-bucket-name--azid--x-s3
```

有关更多信息，请参阅《IAM 用户指南》中的 Amazon Resource Names (ARNs)。有关资源的更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：Resource。

适用于 S3 Express One Zone 的操作

在基于 IAM 身份的策略或基于资源的策略中，您可以定义要允许或拒绝的 S3 操作。S3 Express One Zone 操作对应于特定的 API 操作。S3 Express One Zone 具有唯一的 IAM 命名空间，这不同于 Amazon S3 的标准命名空间。此命名空间是 s3express。

当您允许 s3express:CreateSession 权限时，这会让 CreateSession API 操作能够在访问可用区端点 API（或对象级）操作时检索会话令牌。这些会话令牌返回的凭证用于授予对其他所有可用区端点 API 操作的访问权限。因此，您不必使用 IAM 策略授予对可用区 API 操作的访问权限，而是通过会话令牌来允许访问。

有关可用区和区域端点 API 操作的更多信息，请参阅 S3 Express One Zone 联网。要了解有关 CreateSession API 操作的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CreateSession。

您可以在 IAM 策略语句的 Action 元素中指定以下操作。可以使用策略授予在 AWS 中执行操作的权限。当您在策略中使用一项操作时，通常会允许或拒绝使用具有相同名称的 API 操作。但在某些情况下，单个操作可控制对多个 API 操作的访问。对存储桶级操作的访问权限，只能在基于 IAM 身份的策略（用户或角色）中授予，不能在存储桶策略中授予。

适用于 S3 Express One Zone 的操作和条件键

操作	API	描述	访问级别	条件键
s3express:CreateBucket	CreateBucket	授予权限以创建新的存储桶。	写入	s3express:authType s3express:LocationName s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256
s3express:CreateSession	CreateSession	授予权限以创建会话令牌，该令牌用于授予对所有可用区（对象级）API 操作的访问权限，例如 PutObject 、GetObject 等。	写入	s3express:authType s3express:SessionMode s3express:ResourceAccount

操作	API	描述	访问级别	条件键
				s3express:signatureversion s3express:signatureAge s3express:TlsVersion s3express:x-amz-content-sha256
s3express:DeleteBucket	DeleteBucket	授予权限以删除在 URI 中指定的存储桶。	写入	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

操作	API	描述	访问级别	条件键
s3express:DeleteBucketPolicy	DeleteBucketPolicy	授予权限以删除指定存储桶上的策略。	权限管理	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

操作	API	描述	访问级别	条件键
s3express:GetBucketPolicy	GetBucketPolicy	授予权限以返回指定存储桶的策略。	读取	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

操作	API	描述	访问级别	条件键
s3express:ListAllMyDirectoryBuckets	ListDirectoryBuckets	授予权限，以列出由已通过身份验证的请求发出方拥有的所有目录存储桶。	列出	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

操作	API	描述	访问级别	条件键
s3express:PutBucketPolicy	PutBucketPolicy	授予权限以在存储桶上添加或替换存储桶策略。	权限管理	s3express:authType s3express:ResourceAccount s3express:signatureversion s3express:TlsVersion s3express:x-amz-content-sha256

适用于 S3 Express One Zone 的条件键

S3 Express One Zone 定义了以下可以在 IAM 策略的 Condition 元素中使用的条件键。您可以使用这些键进一步细化应用策略语句的条件。

条件键	描述	类型
s3express:authType	按身份验证方法筛选访问。要限定传入的请求使用特定的身份认证方法，您可以使用此可选条件键。例如，您可以使用此条件键以仅允许在请求身份认证中使用 HTTP Authorization 标头。	字符串

条件键	描述	类型
	有效值：REST-HEADER 、REST-QUERY-STRING	
s3express:LocationName	按特定的可用区 ID（AZ ID）筛选对 CreateBucket API 操作的访问权限，例如 usw2-az1。 示例值：usw2-az1	字符串
s3express:ResourceAccount	按资源拥有者的 AWS 账户 ID 筛选访问权限。 要限制用户、角色或应用程序对特定 AWS 账户 ID 所拥有的目录桶的访问权限，您可以使用 aws:ResourceAccount 或 s3express:ResourceAccount 条件键。您可以在 AWS Identity and Access Management（IAM）身份策略或虚拟私有云（VPC）端点策略中使用此条件键。例如，您可以使用此条件键来限制 VPC 内的客户端访问非您拥有的桶。 示例值：111122223333	字符串
s3express:SessionMode	按照 CreateSession API 操作请求的权限来筛选访问权限。默认情况下，会话为 ReadWrite 。您可以使用此条件键将访问权限限制为 ReadOnly，或者明确拒绝 ReadWrite 访问。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的S3 Express One Zone 目录存储桶策略示例和 CreateSession。 有效值：ReadWrite 、ReadOnly	字符串

条件键	描述	类型
s3express:signatureAge	按请求签名的生存期（以毫秒为单位）筛选访问。此条件仅适用于预签名 URL。 在 AWS 签名版本 4 中，签名密钥的有效期最长为 7 天。因此，签名的有效期最初也为 7 天。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 签名请求简介。您可以使用此条件进一步限制签名有效期。 示例值：600000	数值
s3express:signatureversion	标识您希望支持的 AWS 签名的版本，以将其用于经过身份验证的请求。对于经过身份验证的请求，S3 Express One Zone 支持签名版本 4。 有效值："AWS4-HMAC-SHA256" （标识签名版本 4）	字符串
s3express:TlsVersion	按客户端使用的 TLS 版本筛选访问权限。 您可以使用 s3:TlsVersion 条件键来编写 IAM、虚拟私有云端点（VPCE）或桶策略，以便根据客户端使用的 TLS 版本，限制用户或应用程序对目录桶的访问权限。您还可以使用此条件键来编写具有最低 TLS 版本要求的策略。 示例值：1.3	数值

条件键	描述	类型
s3express:x-amz-content-sha256	按存储桶中未签名内容筛选访问权限。 您可以使用此条件键禁止存储桶中未签名的内容。 使用签名版本 4 时，对于使用 Authorization 标头的请求，您需要在签名计算中添加 x-amz-content-sha256 标头，然后将其值设置为哈希负载。 您可以在存储桶策略中使用此条件键来拒绝上传任何未签名的有效负载。例如： • 拒绝使用 Authorization 标头对请求进行身份认证，但未对有效负载进行签名的上传。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的在单个区块中传输有效负载。 • 拒绝使用预签名 URL 的上传。预签名 URL 始终有一个 UNSIGNED_PAYLOAD 。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的对请求进行身份认证和身份认证方法。 有效值：UNSIGNED-PAYLOAD	字符串

如何对 API 操作进行授权和身份验证

下表列出了 S3 Express One Zone API 操作的授权和身份验证信息。对于每个 API 操作，表中显示了 API 操作名称、IAM 操作、端点类型（区域或可用区）和授权机制（IAM 或基于会话）。此表还指出了哪些区域支持跨账户访问。对存储桶级操作的访问权限，只能在基于 IAM 身份的策略（用户或角色）中授予，不能在存储桶策略中授予。

API	端点类型	IAM 操作	跨账户存取
CreateBucket	区域性	s3express:CreateBucket	不支持
DeleteBucket	区域性	s3express:DeleteBucket	不支持
ListDirectoryBuckets	区域性	s3express:ListAllMyDirectoryBuckets	不支持
PutBucketPolicy	区域性	s3express:PutBucketPolicy	不支持
GetBucketPolicy	区域性	s3express:GetBucketPolicy	不支持
DeleteBucketPolicy	区域性	s3express:DeleteBucketPolicy	不支持
CreateSession	可用区	s3express:CreateSession	支持
CopyObject	可用区	s3express:CreateSession	支持
DeleteObject	可用区	s3express:CreateSession	支持
DeleteObjects	可用区	s3express:CreateSession	支持
HeadObject	可用区	s3express:CreateSession	支持
PutObject	可用区	s3express:CreateSession	支持
GetObjectAttributes	可用区	s3express:CreateSession	支持
ListObjectsV2	可用区	s3express:CreateSession	支持
HeadBucket	可用区	s3express:CreateSession	支持
CreateMultipartUpload	可用区	s3express:CreateSession	支持
UploadPart	可用区	s3express:CreateSession	支持

API	端点类型	IAM 操作	跨账户存取
UploadPartCopy	可用区	s3express:CreateSession	支持
CompleteMultipartUpload	可用区	s3express:CreateSession	支持
AbortMultipartUpload	可用区	s3express:CreateSession	支持
ListParts	可用区	s3express:CreateSession	支持
ListMultipartUploads	可用区	s3express:CreateSession	支持

适用于 S3 Express One Zone 的基于 IAM 身份的策略

在创建目录存储桶或使用 Amazon S3 Express One Zone 存储类之前，您必须向 AWS Identity and Access Management（IAM）角色或用户授予必要的权限。此示例策略允许使用 CreateSession API 操作 [用于可用区端点（对象级）API 操作] 和所有区域端点（存储桶级）API 操作。此策略允许将 CreateSession API 操作用于所有目录存储桶，但仅允许将区域端点 API 操作用于指定的目录存储桶。要使用此示例策略，请将 *user input placeholders* 替换为您自己的信息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccessRegionalEndpointAPIs",
            "Effect": "Allow",
            "Action": [
                "s3express:DeleteBucket",
                "s3express:DeleteBucketPolicy",
                "s3express:CreateBucket",
                "s3express:PutBucketPolicy",
                "s3express:GetBucketPolicy",
                "s3express:ListAllMyDirectoryBuckets"
            ],
```

```
            "Resource": "arn:aws:s3express:region:account_id:bucket/bucket-base-
name--azid--x-s3/*"
        },
        {
            "Sid": "AllowCreateSession",
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "*"
        }
    ]
}
```

S3 Express One Zone 目录存储桶策略示例

本节提供了与 Amazon S3 Express One Zone 存储类配合使用的目录存储桶策略示例。要使用这些策略，请将 *user input placeholders* 替换为您自己的信息。

以下示例存储桶策略允许 AWS 账户 ID *111122223333* 在指定目录存储桶的默认 ReadWrite 会话中使用 CreateSession API 操作。此策略授予对可用区端点（对象级）API 操作的访问权限。

Example – 允许通过默认 **ReadWrite** 会话执行 **CreateSession** 调用的存储桶策略

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccess",
            "Effect": "Allow",
            "Resource": "arn:aws:s3express:us-west-2:account-id:bucket/bucket-base-
name--azid--x-s3",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            },
            "Action": [
                "s3express:CreateSession"
            ]
        }
    ]
}
```

Example – 允许通过 **ReadOnly** 会话执行 **CreateSession** 调用的存储桶策略

以下示例存储桶策略允许 AWS 账户 ID *111122223333* 使用 CreateSession API 操作。此策略使用 s3express:SessionMode 条件键以及 ReadOnly 值来设置只读会话。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express:CreateSession",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3express:SessionMode": "ReadOnly"
                }
            }
        }
    ]
}
```

Example – 允许 **CreateSession** 调用进行跨账户访问的存储桶策略

以下示例存储桶策略允许 AWS 账户 ID *111122223333* 在由 AWS 账户 ID *444455556666* 拥有的指定目录存储桶中，使用 CreateSession API 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccount",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "111122223333"
            },
            "Action": [
                "s3express:CreateSession"
            ],
            "Resource": "arn:aws:s3express:us-west-2:444455556666:bucket/bucket-base-name--azid--x-s3"

        }
    ]
}
```

CreateSession 授权

Amazon S3 Express One Zone 同时支持 AWS Identity and Access Management（AWS IAM）授权和基于会话的授权：

- 要在 S3 Express One Zone 中使用区域端点 API 操作（桶级或控制面板操作），请使用 IAM 授权模型，该模型不涉及会话管理。对于操作，将单独授予权限。有关更多信息，请参阅适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。
- 要使用可用区端点 API 操作（对象级或数据面板操作），您可以使用 CreateSession API 操作来创建和管理会话，这些会话经过优化，为数据请求提供低延迟的授权。要检索和使用会话令牌，您必须在基于身份的策略或存储桶策略中，允许对目录存储桶执行 s3express:CreateSession 操作。有关更多信息，请参阅适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。如果您在 Amazon S3 控制台中通过 AWS Command Line Interface（AWS CLI）或使用 AWS SDK 访问 S3 Express One Zone，S3 Express One Zone 会代表您创建会话。

如果您使用 Amazon S3 REST API，则可以使用 CreateSession API 操作获取临时安全凭证，其中包括访问密钥 ID、秘密访问密钥、会话令牌和到期时间。临时凭证提供的权限与长期安全凭证（例如 IAM 用户凭证）相同，但临时安全凭证必须包括会话令牌。

会话模式

会话模式定义会话的范围。在存储桶策略中，您可以指定 s3express:SessionMode 条件密钥，以便控制谁可以创建 ReadWrite 或 ReadOnly 会话。有关 ReadWrite 或 ReadOnly 会话的更多信息，请参阅《Amazon S3 API 参考》中 CreateSession 的 x-amz-create-session-mode 参数。有关要创建的存储桶策略的更多信息，请参阅 S3 Express One Zone 目录存储桶策略示例。

会话令牌

当您使用临时安全凭证进行调用时，调用必须包含会话令牌。会话令牌与临时凭证一起返回。会话令牌的范围仅限于您的目录存储桶，用于验证安全凭证是否有效且未过期。为了保护您的会话，临时安全凭证在 5 分钟后过期。

`CopyObject` 和 `HeadBucket`

临时安全凭证的范围限定为特定的目录存储桶，并且对于发送到给定目录存储桶的所有可用区（对象级）操作 API 调用，将自动启用临时安全凭证。与其他可用区端点 API 操作不同，`CopyObject` 和 `HeadBucket` 不使用 `CreateSession` 身份验证。所有 `CopyObject` 和 `HeadBucket` 请求都必须使用 IAM 凭证进行身份验证和签名。但是，`CopyObject` 和 `HeadBucket` 与其他可用区端点 API 操作一样，仍由 `s3express:CreateSession` 授权。

有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CreateSession。

S3 Express One Zone 的安全最佳实践

Amazon S3 提供了大量安全功能，您在开发和实施自己的安全策略时可以考虑使用。以下最佳实操是一般准则，并不代表完整的安全解决方案。这些最佳实践可能不适合您的环境或不满足您的环境要求，请将其视为有用的建议而不是惯例。

默认屏蔽公共访问权限和对象所有权设置

要使用 S3 Express One Zone 存储类，您必须使用 S3 目录存储桶。目录存储桶支持 S3 屏蔽公共访问权限和 S3 对象所有权。这些 S3 功能用于审计和管理对存储桶及对象的访问。

默认情况下，所有的目录存储桶都启用了屏蔽公共访问权限设置。此外，对象所有权设置为强制存储桶拥有者，这意味着访问控制列表（ACL）已禁用。无法修改这些设置。有关使用这些功能的更多信息，请参阅the section called “阻止公有访问”和the section called “控制对象所有权”。

> **Note**
>
> 您无法授予对存储在目录存储桶中的对象的访问权限。您只能授予对目录存储桶的访问权限。S3 Express One Zone 的授权模式与 Amazon S3 的授权模式不同。有关更多信息，请参阅 `CreateSession` 授权。

身份验证和授权

根据您是向可用区端点 API 操作发出请求还是向区域端点 API 操作发出请求，S3 Express One Zone 的身份验证和授权机制会有所不同。可用区 API 操作是对象级（数据面板）操作。区域 API 操作是存储桶级（控制面板）操作。

借助 S3 Express One Zone，您可以通过基于会话的全新机制，对针对可用区端点 API 操作的请求进行身份验证和授权，该机制经过优化，可提供极低的延迟。通过基于会话的身份验证，AWS SDK 使用 `CreateSession` API 操作来请求临时凭证，以提供对目录存储桶的低延迟访问。这些临时凭证的作用范围限制为特定的目录存储桶，并在 5 分钟后过期。您可以使用这些临时凭证对可用区（对象级）API 调用签名。有关更多信息，请参阅 `CreateSession` 授权。

使用 S3 Express One Zone 凭证对请求签名

您可以通过您的 S3 Express One Zone 凭证，使用 AWS 签名版本 4 并将 `s3express` 作为服务名称，对可用区端点（对象级）API 请求进行签名。在对请求签名时，请使用从 `CreateSession` 返回的密钥，还要通过 `x-amzn-s3session-token header` 提供会话令牌。有关更多信息，请参阅 CreateSession。

适用于 S3 Express One Zone 类的支持的 AWS SDK 负责管理凭证和签名事宜。我们建议使用适用于 S3 Express One Zone 的 AWS SDK 来刷新凭证并为您签名请求。

使用 IAM 凭证对请求签名

所有区域（存储桶级）API 调用都必须通过 AWS Identity and Access Management（IAM）凭证（而不是临时会话凭证）进行身份验证和签名。IAM 凭证由 IAM 身份的访问密钥 ID 和秘密访问密钥组成。所有 `CopyObject` 和 `HeadBucket` 请求还必须使用 IAM 凭证进行身份验证和签名。

为了尽可能减少可用区（对象级）操作调用的延迟，我们建议使用通过调用 `CreateSession` 获得的 S3 Express One Zone 凭证来对您的请求签名，但 `CopyObject` 和 `HeadBucket` 的请求除外。

使用 AWS CloudTrail

AWS CloudTrail 提供用户、角色或 AWS 服务在 Amazon S3 中执行的操作的记录。您可以使用 CloudTrail 收集的信息确定以下事项：

- 向 Amazon S3 发出的请求
- 发出请求的 IP 地址
- 谁发出了请求
- 发出请求的时间
- 有关该请求的其他详细信息

当您设置 AWS 账户时，CloudTrail 管理事件默认处于启用状态。以下区域端点 API 操作（存储桶级或控制面板 API 操作）将记录到 CloudTrail 中。

- CreateBucket

- DeleteBucket
- DeleteBucketPolicy
- PutBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- ListMultipartUploads

> **Note**
>
> `ListMultipartUploads` 是可用区端点 API 操作。但是，它会作为管理事件记录到 CloudTrail 中。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 ListMultipartUploads。

默认情况下，CloudTrail 跟踪不记录数据事件，但您可以将跟踪配置为记录您指定的目录存储桶的数据事件，或记录您的 AWS 账户中所有目录存储桶的数据事件。以下可用区端点 API 操作（对象级或数据面板 API 操作）将记录到 CloudTrail 中。

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateSession
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- ListParts
- PutObject
- UploadPart

- UploadPartCopy

有关将 AWS CloudTrail 与 S3 Express One Zone 结合使用的更多信息，请参阅使用 AWS CloudTrail 为 S3 Express One Zone 记录日志。

使用 AWS 监控工具实施监控

监控是保持 Amazon S3 和您的 AWS 解决方案的可靠性、安全性、可用性和性能的重要部分。AWS 提供了一些可用来监控 Amazon S3 和您的其他 AWS 服务的工具和服务。例如，您可以监控 Amazon S3 的 Amazon CloudWatch 指标，特别是 `BucketSizeBytes` 和 `NumberOfObjects` 存储指标。

存储在 S3 Express One Zone 存储类中的对象不会反映在 Amazon S3 的 `BucketSizeBytes` 和 `NumberOfObjects` 存储指标中。但是，S3 Express One Zone 支持 `BucketSizeBytes` 和 `NumberOfObjects` 存储指标。要查看您选择的指标，您可以通过指定 `StorageType` 维度来区分 Amazon S3 存储类和 S3 Express One Zone 存储类。有关更多信息，请参阅 使用 Amazon CloudWatch 监控指标。

有关更多信息，请参阅使用 Amazon CloudWatch 监控指标 和监控 Amazon S3。

使用 AWS CloudTrail 为 S3 Express One Zone 记录日志

Amazon S3 与 AWS CloudTrail 集成，后者是一项提供由用户、角色或 AWS 服务所采取操作的记录的服务。CloudTrail 将 Amazon S3 的所有 API 调用作为事件捕获。借助通过 CloudTrail 收集的信息，您可以确定向 Amazon S3 发出哪些请求、发出请求的 IP 地址、请求的发出时间以及其它详细信息。当 Amazon S3 中发生受支持的事件活动时，该活动将记录在 CloudTrail 事件中。可以使用 AWS CloudTrail 跟踪来记录 S3 Express One Zone 的管理事件和数据事件。有关更多信息，请参阅 Amazon S3 CloudTrail 事件和《AWS CloudTrail 用户指南》中的 What is AWS CloudTrail?

S3 Express One Zone 的 CloudTrail 管理事件

默认情况下，CloudTrail 将目录存储桶的存储桶级操作记录为管理事件。S3 Express One Zone 的 CloudTrail 管理事件的 `eventsource` 为 `s3express.amazonaws.com`。当您设置 AWS 账户时，CloudTrail 管理事件默认处于启用状态。以下区域端点 API 操作（存储桶级或控制面板 API 操作）将记录到 CloudTrail 中。

- `CreateBucket`
- `DeleteBucket`
- `DeleteBucketPolicy`

- PutBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- ListMultipartUploads

> **Note**
>
> ListMultipartUploads 是可用区端点 API 操作。但是，此 API 操作会作为管理事件记录到 CloudTrail 中。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 ListMultipartUploads。

有关 CloudTrail 管理事件的更多信息，请参阅《AWS CloudTrail 用户指南》中的 Logging management events。

S3 Express One Zone 的 CloudTrail 数据事件

数据事件可提供对资源或在资源中所执行资源操作（例如，读取或写入 Amazon S3 对象）的相关信息。这些也称为数据层面操作。数据事件通常是高容量活动。默认情况下，CloudTrail 跟踪不记录数据事件，但您可以将跟踪配置为记录存储在通用存储桶和目录存储桶中的对象的数据事件。有关更多信息，请参阅使用控制台为存储桶中的对象启用日志记录功能。

在 CloudTrail 中记录跟踪的数据事件时，您可以选择使用高级事件选择器或基本事件选择器。要记录存储在目录存储桶中的对象的数据事件，必须使用高级事件选择器。配置高级资源选择器时，您将需要为 S3 Express One Zone 选择或指定资源类型，即 AWS::S3Express::Object。

以下可用区端点 API 操作（对象级或数据面板 API 操作）将记录到 CloudTrail 中。

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateSession
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- GetObject

- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- ListParts
- PutObject
- UploadPart
- UploadPartCopy

有关 CloudTrail 数据事件的更多信息，请参阅《AWS CloudTrail 用户指南》中的 Logging data events。

有关 S3 Express One Zone 的 CloudTrail 事件的更多信息，请参阅以下主题：

主题

- S3 Express One Zone 的 CloudTrail 日志文件示例

S3 Express One Zone 的 CloudTrail 日志文件示例

CloudTrail 日志文件包括有关所请求 API 操作的信息、操作的日期和时间和请求参数等。本主题介绍 S3 Express One Zone 的 CloudTrail 数据事件和管理事件示例。

主题

- Amazon S3 Express One Zone 存储类的 CloudTrail 数据事件日志文件示例

Amazon S3 Express One Zone 存储类的 CloudTrail 数据事件日志文件示例

下面的示例显示了一个 CloudTrail 日志文件示例，该示例演示了 CreateSession。

```
{
    "eventVersion": "1.09",
    "userIdentity": {
      "type": "AssumedRole",
      "principalId": "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName",
      "arn": "arn:aws:sts::111122223333assumed-role/RoleToBeAssumed/MySessionName",
      "accountId": "111122223333",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
```

```
          "sessionContext": {
            "sessionIssuer": {
              "type": "Role",
              "principalId": "AROAIDPPEZS35WEXAMPLE",
              "arn": "arn:aws:iam::111122223333:role/RoleToBeAssumed",
              "accountId": "111122223333",
              "userName":"RoleToBeAssumed
            },

            "attributes": {
              "creationDate": "2024-07-02T00:21:16Z",
            "mfaAuthenticated": "false"
            }
          }
        },
        "eventTime": "2024-07-02T00:22:11Z",
        "eventSource": "s3express.amazonaws.com",
        "eventName": "CreateSession",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "72.21.198.68",
        "userAgent": "aws-sdk-java/2.20.160-SNAPSHOT
 Linux/5.10.216-225.855.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/11.0.23+9-LTS
 Java/11.0.23 vendor/Amazon.com_Inc. md/internal exec-env/AWS_Lambda_java11 io/sync
 http/Apache cfg/retry-mode/standard",
        "requestParameters": {
          "bucketName": "bucket-base-name--usw2-az1--x-s3".
            "host": "bucket-base-name--usw2-az1--x-s3.s3express-usw2-az1.us-
west-2.amazonaws.com",
            "x-amz-create-session-mode": "ReadWrite"
        },
        "responseElements": {
            "credentials": {
                "accessKeyId": "AKIAI44QH8DHBEXAMPLE"
                "expiration": ""Mar 20, 2024, 11:16:09 PM",
                "sessionToken": "<session token string>"
           },
        },
        "additionalEventData": {
            "SignatureVersion": "SigV4",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "bytesTransferredIn": 0,
            "AuthenticationMethod": "AuthHeader",
            "xAmzId2": "q6xhNJYmhg",
            "bytesTransferredOut": 1815,
```

```
            "availabilityZone": "usw2-az1"
          },
          "requestID": "28d2faaf-3319-4649-998d-EXAMPLE72818",
          "eventID": "694d604a-d190-4470-8dd1-EXAMPLEe20c1",
          "readOnly": true,
          "resources": [
            {
              "type": "AWS::S3Express::Object",
              "ARNPrefix": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--
usw2-az1--x-s3"
            },
            {
              "accountId": "111122223333"
              "type": "AWS::S3Express::DirectoryBucket",
              "ARN": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--usw2-
az1--x-s3"
             }
           ],
           "eventType": "AwsApiCall",
           "managementEvent": false,
           "recipientAccountId": "111122223333",
           "eventCategory": "Data",
           "tlsDetails": {
             "tlsVersion": "TLSv1.3",
             "cipherSuite": "TLS_AES_128_GCM_SHA256",
             "clientProvidedHostHeader": "bucket-base-name--usw2-az1--x-s3.s3express-
usw2-az1.us-west-2.amazonaws.com"
            }
          }
```

要使用可用区端点 API 操作（对象级或数据面板操作），可以使用 CreateSession API 操作来创建和管理会话，这些会话经过优化，可为数据请求提供低延迟的授权。也可以使用 CreateSession 来减少日志记录量。要确定在会话期间执行了哪些可用区 API 操作，可以将 CreateSession 日志文件中 responseElements 下的 accessKeyId 与其它可用区 API 操作的日志文件中的 accessKeyId 进行匹配。有关更多信息，请参阅 CreateSession 授权。

下面的示例显示了一个 CloudTrail 日志文件示例，该示例演示了通过 CreateSession 进行身份验证的 GetObject API 操作。

```
    {
        "eventVersion": "1.09",
        "userIdentity": {
```

```
          "type": "AssumedRole",
          "principalId": "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName",
          "arn": "arn:aws:sts::111122223333assumed-role/RoleToBeAssumed/MySessionName",
          "accountId": "111122223333",
          "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
          "sessionContext": {
            "attributes": {
              "creationDate": "2024-07-02T00:21:49Z"
            }
          }
        },
        "eventTime": "2024-07-02T00:22:01Z",
        "eventSource": "s3express.amazonaws.com",
        "eventName": "GetObject",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "72.21.198.68",
        "userAgent": "aws-sdk-java/2.25.66 Linux/5.10.216-225.855.amzn2.x86_64
 OpenJDK_64-Bit_Server_VM/17.0.11+9-LTS Java/17.0.11 vendor/Amazon.com_Inc. md/internal
 exec-env/AWS_Lambda_java17 io/sync http/Apache cfg/retry-mode/legacy",
        "requestParameters": {
          "bucketName": "bucket-base-name--usw2-az1--x-s3",
          "x-amz-checksum-mode": "ENABLED",
          "Host": "bucket-base-name--usw2-az1--x-s3.s3express-usw2-az1.us-
west-2.amazonaws.com",
          "key": "test-get-obj-with-checksum"
        },
        "responseElements": null,
        "additionalEventData": {
          "SignatureVersion": "Sigv4",
          "CipherSuite": "TLS_AES_128_GCM_SHA256",
          "bytesTransferredIn": 0,
          "AuthenticationMethod": "AuthHeader",
          "x-amz-id-2": "oOy6w8K7LFsyFN",
          "bytesTransferredOut": 9,
          "availabilityZone": "usw2-az1",
          "sessionModeApplied": "ReadWrite"
         },
          "requestID": "28d2faaf-3319-4649-998d-EXAMPLE72818",
          "eventID": "694d604a-d190-4470-8dd1-EXAMPLEe20c1",
          "readOnly": true,
          "resources": [
            {
              "type": "AWS::S3Express::Object",
```

```
              "ARNPrefix": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--
usw2-az1--x-s3"
            },
            {
              "accountId": "111122223333",
              "type": "AWS::S3Express::DirectoryBucket",
              "ARN": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--usw2-
az1--x-s3"
             }
           ],
           "eventType": "AwsApiCall",
           "managementEvent": false,
           "recipientAccountId": "111122223333",
           "eventCategory": "Data",
           "tlsDetails": {
             "tlsVersion": "TLSv1.3",
             "cipherSuite": "TLS_AES_128_GCM_SHA256",
             "clientProvidedHostHeader": "bucket-base-name--usw2-az1--x-s3.s3express-
usw2-az1.us-west-2.amazonaws.com"
            }
          }
```

在上面的 GetObject 日志文件示例中，accessKeyId（AKIAI44QH8DHBEXAMPLE）与 CreateSession 日志文件示例中 responseElements 下的 accessKeyId 相匹配。匹配的 accessKeyId 表示在哪个会话中执行了 GetObject 操作。

优化 Amazon S3 Express One Zone 性能

Amazon S3 Express One Zone 是高性能的单可用区（AZ）S3 存储类，专门用于为注重延迟的应用程序提供稳定的毫秒级数据访问。S3 Express One Zone 是第一种允许您选择将高性能对象存储和 AWS 计算资源（例如 Amazon Elastic Compute Cloud、Amazon Elastic Kubernetes Service 和 Amazon Elastic Container Service）联合托管在单个可用区内的 S3 存储类。将存储和计算资源联合托管在一起，可以优化计算性能和成本，并提高数据处理速度。

S3 Express One Zone 提供了与其他 S3 存储类相似的性能弹性，但首字节读取和写入请求延迟能够稳定地保持在毫秒级，比 S3 Standard 存储快 10 倍。S3 Express One Zone 采用了全新设计，可支持非常高聚合水平的突发吞吐量。S3 Express One Zone 存储类使用定制架构来优化性能，并通过将数据存储在高性能硬件中，提供稳定的低请求延迟。S3 Express One Zone 的对象协议进行了增强，用于简化身份验证和元数据开销。

为了进一步提高访问速度并支持每秒数十万个请求，S3 Express One Zone 将数据存储在新的存储桶类型中：Amazon S3 目录存储桶。每个 S3 目录存储桶均可支持数十万的每秒事务数（TPS）。

S3 Express One Zone 将提供数据访问速度达毫秒级的高性能专用硬件和软件，与可进行扩展以便每秒处理大量事务的目录存储桶相结合，使其成为非常适合请求密集型操作或注重性能的应用程序的 Amazon S3 存储类。

以下主题针对使用 S3 Express One Zone 存储类的应用程序，介绍了用于优化性能的最佳实践准则和设计模式。

主题

S3 Express One Zone 的性能准则和设计模式

构建从 Amazon S3 Express One Zone 上传和检索对象的应用程序时，请遵循最佳实践准则以优化性能。要使用 S3 Express One Zone 存储类，您必须创建 S3 目录存储桶。S3 Express One Zone 存储类不支持与 S3 通用存储桶一起使用。

有关其他所有 Amazon S3 存储类和 S3 通用存储桶的性能准则，请参阅最佳实践设计模式：优化 Amazon S3 性能。

应用程序在使用 S3 Express One Zone 存储类和目录存储桶时，要想获得最佳性能，我们建议遵循以下准则和设计模式。

主题

将 S3 Express One Zone 存储与您的 AWS 计算资源联合托管在一个位置

每个目录存储桶存储在创建存储桶时选择的单个可用区中。首先，在计算工作负载或资源所在的可用区中，可以创建一个新的目录存储桶。然后就可以立即开始享受到延迟非常低的读取和写入。目录存储桶是第一种可以选择 AWS 区域中可用区的 S3 存储桶，可以减少计算和存储之间的延迟。

如果您跨可用区访问目录存储桶，则延迟会增加。为了优化性能，我们建议您尽可能从位于同一可用区的 Amazon Elastic Container Service、Amazon Elastic Kubernetes Service 和 Amazon Elastic Compute Cloud 实例访问目录存储桶。

目录桶

每个目录存储桶均可支持数十万的每秒事务数（TPS）。与通用存储桶不同，目录存储桶将键分层组织到目录中，而不是前缀中。前缀是对象键名称开头的一串字符串。您可以把前缀视为一种以类似于目录的方式组织数据的方式。但是，前缀不是目录。

前缀在通用存储桶的平面命名空间中组织数据，通用存储桶中的前缀数量没有限制。每个前缀每秒可以实现每秒至少 3500 个 PUT/POST/DELETE 或 5500 个 GET/HEAD 请求。您还可以并行处理多个前缀的请求以扩展性能。但是，无论是读取操作还是写入操作，此扩展都是逐渐发生的，而不是瞬间发生。当通用存储桶扩展到新的更高请求速率时，您可能会收到一些 HTTP 状态码 503（服务不可用）错误。

对于分层命名空间，对象键中的分隔符很重要。唯一支持的分隔符是正斜杠（`/`）。目录由分隔符边界确定。例如，对象键 `dir1/dir2/file1.txt` 会自动创建目录 `dir1/` 和 `dir2/`，并将对象 `file1.txt` 添加到路径 `dir1/dir2/file1.txt` 中的 `/dir2` 目录。

将对象上传到目录存储桶时创建的目录对每个前缀没有 TPS 限制，并且会自动进行预扩展，以减少出现 HTTP 503（服务不可用）错误的可能性。通过这种自动扩展，您的应用程序可以根据需要并行处理目录内和跨目录的读取和写入请求。

目录存储桶横向扩展请求并行化

您可以将多个并行请求发送到目录存储桶，以便在不同的连接上分布请求，尽可能充分利用可用带宽，从而实现出色的性能。S3 Express One Zone 对与目录存储桶建立的连接数没有任何限制。在对同一目录进行大量并发写入操作时，这一目录可以横向自动扩展性能。

在最初创建对象键并且其键名包含目录时，将自动为该对象创建该目录。后续对象上传到相同目录时不需要创建目录，这样可以减少将对象上传到现有目录的延迟。

虽然目录存储桶中支持以浅层目录结构和深层目录结构存储对象，但目录存储桶可以自动横向扩展，同时上传到相同目录或并行同级目录的延迟更低。

使用基于会话的身份验证

S3 Express One Zone 和目录存储桶支持基于会话的新型授权机制，用于对发送到目录存储桶的请求进行身份验证和授权。使用基于会话的身份验证，AWS SDK 会自动使用 `CreateSession` API 操作创建临时会话令牌，该令牌可用于对发送到目录存储桶的数据请求进行低延迟授权。

AWS SDK 使用 `CreateSession` API 操作请求临时凭证，然后代表您每 5 分钟自动创建和刷新令牌。为了充分利用 S3 Express One Zone 存储类的性能优势，我们建议您使用 AWS SDK 来启动和管理 `CreateSession` API 请求。有关此基于会话的模型的更多信息，请参阅 `CreateSession` 授权。

其他 S3 校验和最佳实践

S3 Express One Zone 可让您选择用于在上传或下载过程中验证数据的校验和算法。您可以选择以下安全哈希算法（SHA）或循环冗余校验（CRC）数据完整性检查算法之一：CRC32、CRC32C、SHA-1 和 SHA-256。S3 Express One Zone 存储类不支持基于 MD5 的校验和。

CRC32 是 AWS SDK 在对 S3 Express One Zone 往返传输数据时默认使用的校验和。我们建议使用 CRC32 和 CRC32C，以便在 S3 Express One Zone 存储类上获得最佳性能。

使用最新版本的 AWS SDK 和通用运行时库

一些 AWS SDK 还提供 AWS 通用运行时（CRT）库，以进一步提高 S3 客户端的性能。这些 SDK 包括 AWS SDK for Java 2.x、AWS SDK for C++ 和 AWS SDK for Python (Boto3)。基于 CRT 的 S3 客户端通过自动使用分段上传 API 操作和字节范围提取来自动进行横向扩展连接，从而以高性能和可靠性向 S3 Express One Zone 往返传输数据。

要使用 S3 Express One Zone 存储类实现极佳的性能，我们建议使用包含 CRT 库的 AWS SDK 的最新版本使用 AWS Command Line Interface（AWS CLI）。

性能故障排除

延迟敏感型应用程序的重试请求

S3 Express One Zone 专为提供稳定的高性能而设计，无需额外调整。但是，设置更主动的超时值和重试次数可以进一步推动实现稳定的延迟和性能。AWS SDK 具有可配置的超时和重试值，您可以进行调整以符合特定应用程序的容限。

AWS Common Runtime（CRT）库和 Amazon EC2 实例类型搭配

执行大量读取和写入操作的应用程序，相比不执行这些操作的应用程序会需要更多的内存或计算容量。在为具有高性能要求的工作负载启动 Amazon Elastic Compute Cloud（Amazon EC2）实例时，

应选择具有您的应用程序需要的这些资源量的实例类型。S3 Express One Zone 高性能存储非常适合与更大、更新的实例类型搭配使用，这些实例类型具有更大的系统内存以及计算能力更强的 CPU 和 GPU，可以利用性能更高的存储。我们还建议使用启用了 CRT 的 AWS SDK 的最新版本，这样可以更好地加速并行读取和写入请求。

在 AWS SDK 而不是 HTTP REST API 中使用基于会话的身份验证

借助 Amazon S3，在使用 HTTP REST API 请求时，您还可以遵循 AWS SDK 中相同的最佳实践，以此来优化性能。但是，对于 S3 Express One Zone 使用的基于会话的授权和身份验证机制，我们强烈建议您使用 AWS SDK 管理 `CreateSession` 及其托管会话令牌。AWS SDK 使用 `CreateSession` API 操作，自动代表您创建和刷新令牌。使用 `CreateSession` 可减少 AWS Identity and Access Management（IAM）授权各个请求的请求往返延迟。

使用 S3 Express One Zone 进行开发

Amazon S3 Express One Zone 是第一种可以在其中选择单个可用区的 S3 存储类，您可以选择将您的对象存储与计算资源联合托管在一个位置，从而提供尽可能高的访问速度。通过 S3 Express One Zone 存储类，您可以使用 S3 目录存储桶来存储数据。每个目录存储桶使用 S3 Express One Zone 存储类在单个可用区中存储对象，您可以在创建存储桶时选择该可用区。

创建目录存储桶后，您便可立即开始实现非常低延迟的读取和写入。您可以通过虚拟私有云（VPC），使用端点连接与目录存储桶进行通信，也可以使用可用区和区域 API 操作管理对象和目录存储桶。您还可以通过 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、AWS SDK 和 Amazon S3 REST API 使用 S3 Express One Zone 存储类。

Amazon S3 Express One Zone 存储类设计为在单个可用区内提供 99.95% 的可用性，并由 Amazon S3 服务等级协议提供保障。使用 S3 Express One Zone，您的数据将冗余地存储在单个可用区中的多个设备上。S3 Express One Zone 设计为通过快速检测和修复任何丢失的冗余来处理并发设备故障。在现有设备出现故障时，S3 Express One Zone 会自动将请求转移到相同可用区内的新设备。这种冗余有助于确保不间断地访问可用区中的数据。

主题

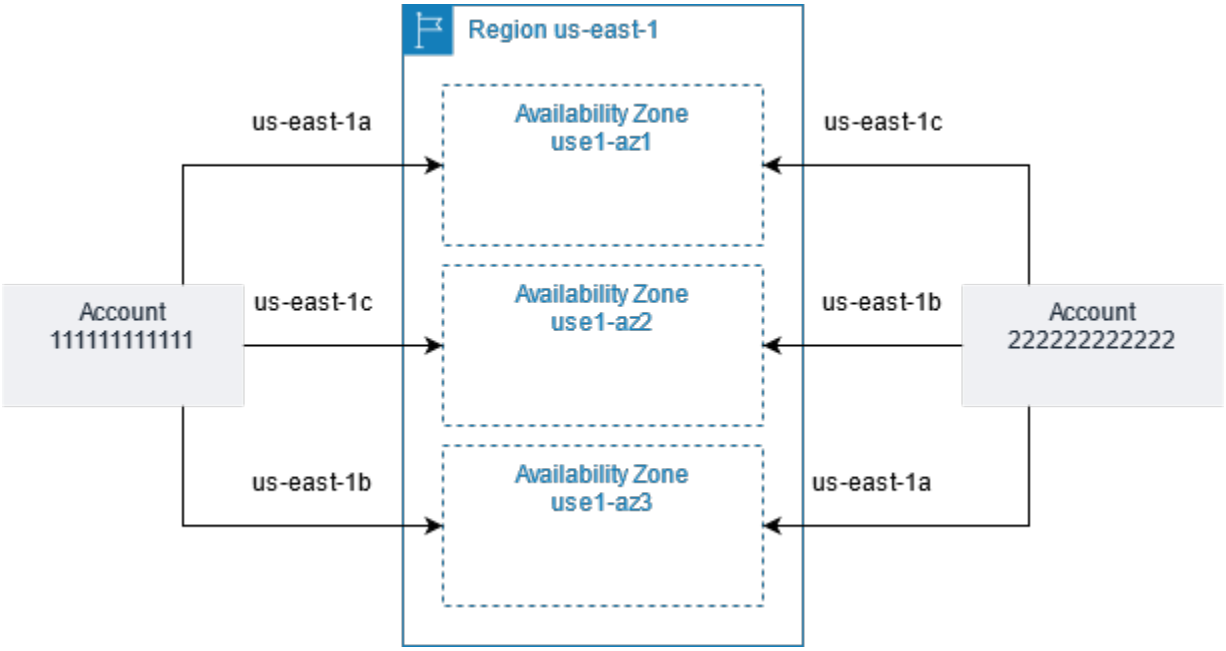
- S3 Express One Zone 可用区和区域
- 区域端点和可用区端点
- 通过 S3 控制台、AWS CLI 和 AWS SDK 使用 S3 Express One Zone
- S3 Express One Zone API 操作

S3 Express One Zone 可用区和区域

可用区是 AWS 区域 中一个或多个具有冗余电源、网络和连接的离散数据中心。为了实现优化的低延迟检索，Amazon S3 Express One Zone 存储类中的对象以冗余方式存储在 S3 目录存储桶中，这种存储桶位于计算工作负载所在的单个可用区。创建目录存储桶时，您可以选择存储桶所在的可用区以及 AWS 区域。

AWS 将物理可用区随机映射到每个 AWS 账户的可用区名称。这种方法有助于在 AWS 区域的可用区之间分配资源，而不是将资源集中在每个区域的第一个可用区中。因此，您的 AWS 账户的可用区 `us-east-1a` 可能与其他 AWS 账户的 `us-east-1a` 表示的物理位置不同。有关更多信息，请参阅《Amazon EC2 用户指南》中的区域和可用区。

要跨账户协调可用区，您必须使用 AZ ID（可用区的唯一、一致的标识符）。例如，`use1-az1` 是 `us-east-1` 区域的可用区 ID，它在每个 AWS 账户 中的实际位置均相同。下图显示了每个账户的 AZ ID 是相同的，尽管每个账户的可用区名称映射可能不同。



使用 S3 Express One Zone，您的数据将冗余地存储在单个可用区中的多个设备上。S3 Express One Zone 设计为在单个可用区内提供 99.95% 的可用性，并由 Amazon S3 服务等级协议提供支持。有关更多信息，请参阅 单可用区

以下区域和可用区支持 S3 Express One Zone：

可使用 S3 Express One Zone 的区域和可用区

区域名称	区域代码	可用区 ID
美国东部（弗吉尼亚州北部）	us-east-1	use1-az4
		use1-az5
		use1-az6
美国西部（俄勒冈）	us-west-2	usw2-az1
		usw2-az3
		usw2-az4
亚太地区（东京）	ap-northeast-1	apne1-az1
		apne1-az4
欧洲地区（斯德哥尔摩）	eu-north-1	eun1-az1
		eun1-az2
		eun1-az3

区域端点和可用区端点

要从虚拟私有云（VPC）中访问 Amazon S3 Express One Zone 的区域端点和可用区端点，您可以使用网关 VPC 端点。创建网关端点后，您可以将其添加作为路由表中的目标，用于从您的 VPC 流向 S3 Express One Zone 的流量。使用网关端点不会发生任何额外费用。有关如何配置网关 VPC 端点的更多信息，请参阅 S3 Express One Zone 联网。

当您使用 S3 Express One Zone 时，存储桶级（控制面板）API 操作可通过区域端点使用，这些操作称为区域端点 API 操作。区域端点 API 操作的示例包括 `CreateBucket` 和 `DeleteBucket`。

创建目录桶后，您可以使用可用区（对象级，或数据面板端点 API 操作）来上传和管理目录桶中的对象。可用区端点 API 操作可通过可用区端点执行。可用区 API 操作的示例包括 `PutObject` 和 `CopyObject`。

通过 S3 控制台、AWS CLI 和 AWS SDK 使用 S3 Express One Zone

您可以通过 AWS SDK、Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 Amazon S3 REST API 使用 S3 Express One Zone 存储类和目录存储桶。

S3 控制台

要开始使用 S3 控制台，请执行以下步骤：

- 创建目录存储桶
- 清空目录存储桶
- 删除目录存储桶

有关完整教程，请参阅教程：开始使用 S3 Express One Zone。

AWS SDK

S3 Express One Zone 支持以下 AWS SDK：

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java 2.x
- AWS SDK for JavaScript v3
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby
- AWS SDK for Kotlin
- AWS SDK for Rust

在使用 S3 Express One Zone 时，建议您使用最新版本的 AWS SDK。S3 Express One Zone 支持的 AWS SDK 代表您处理会话的建立、刷新和终止。这意味着您在下载和安装 AWS SDK 并配置必要的 IAM 权限后，便可立即开始使用 API 操作。有关更多信息，请参阅 适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

有关 AWS SDK 的信息（包括如何下载及安装），请参阅在 AWS 上构建所用的工具。

有关 AWS SDK 示例，请参阅以下内容：

- 创建目录存储桶
- 清空目录存储桶
- 删除目录存储桶

AWS Command Line Interface (AWS CLI)

您可以使用 AWS Command Line Interface（AWS CLI）创建目录存储桶，并对 S3 Express One Zone 使用支持的区域端点和可用区端点 API 操作。

要开始使用 AWS CLI，请参阅《AWS CLI 命令参考》中的开始使用 AWS CLI。

> **Note**
>
> 要通过高级 `aws s3` 命令使用目录存储桶，请将您的 AWS CLI 更新到最新版本。有关如何安装和配置 AWS CLI 的更多信息，请参阅《AWS CLI 用户指南》中的安装或更新 AWS CLI 的最新版本。

有关 AWS CLI 示例，请参阅以下内容：

- 创建目录存储桶
- 清空目录存储桶
- 删除目录存储桶

S3 Express One Zone API 操作

Amazon S3 Express One Zone 存储类同时支持区域（存储桶级或控制面板）和可用区（对象级或数据面板）端点 API 操作。有关更多信息，请参阅S3 Express One Zone 联网和端点和网关 VPC 端点。

区域端点 API 操作

S3 Express One Zone 支持以下区域端点 API 操作：

- CreateBucket

- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- PutBucketPolicy

可用区端点 API 操作

S3 Express One Zone 支持以下可用区端点 API 操作：

- CreateSession
- CopyObject
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject
- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload
- ListMultipartUploads
- ListParts
- UploadPart
- UploadPartCopy

使用 Amazon S3 接入点管理数据访问

Amazon S3 接入点简化了在 S3 中存储数据的任何AWS服务或客户应用程序的数据访问。接入点是附加到存储桶的命名网络端点，您可以使用这些存储桶执行 S3 对象操作（如 `GetObject` 和 `PutObject`）。每个接入点都具有不同的权限和网络控制，S3 将它们应用于通过该接入点发出的任何请求。每个接入点强制实施自定义接入点策略，该策略与附加到底层存储桶的存储桶策略结合使用。您可以将任何接入点配置为仅接受来自 Virtual Private Cloud（VPC）的请求，以限制专用网络的 Amazon S3 数据访问。您还可以为每个接入点配置自定义屏蔽公共访问权限设置。

> **Note**
>
> - 您只能使用接入点来执行对象操作。您不能使用接入点执行其他 Amazon S3 操作，例如修改或删除存储桶。有关支持接入点的 S3 操作的完整列表，请参阅接入点与AWS服务的兼容性。
> - 接入点适用于部分服务和特征，但并不适用于全部 AWS 服务和特征。例如，您不能将跨区域复制配置为通过接入点进行操作。有关与 S3 接入点兼容的 AWS 服务的完整列表，请参阅接入点与AWS服务的兼容性。

本节介绍如何使用 Amazon S3 接入点。有关使用存储桶的信息，请参阅 存储桶概述。有关使用对象的信息，请参阅 Amazon S3 对象概述。

主题

配置使用接入点的 IAM 策略

Amazon S3 接入点支持 AWS Identity and Access Management（IAM）资源策略，这些策略允许您按资源、用户或其他条件控制接入点的使用。要使应用程序或用户能够通过接入点访问对象，接入点和底层存储桶都必须允许请求。

> **Important**
>
> 当通过存储桶的名称或 Amazon 资源名称（ARN）直接访问存储桶时，向存储桶添加 S3 接入点不会改变存储桶的行为。针对存储桶的所有现有操作将继续像以前一样运行。您在接入点策略中包括的限制仅适用于通过该接入点发出的请求。

当您使用 IAM 资源策略时，确保在保存策略之前解决来自 AWS Identity and Access Management Access Analyzer 的安全警告、错误、一般警告和建议。IAM Access Analyzer 将根据 IAM 策略语法和最佳实践运行策略检查，以验证您的策略。这些检查项生成调查结果并提供建议，可帮助您编写可操作且符合安全最佳实践的策略。

要了解有关使用 IAM Access Analyzer 验证策略的更多信息，请参阅《IAM 用户指南》中的 IAM Access Analyzer 策略验证。要查看 IAM Access Analyzer 返回的警告、错误和建议的列表，请参阅 IAM Access Analyzer 策略检查引用。

接入点策略示例

以下示例演示如何创建 IAM 策略来控制通过接入点发出的请求。

> **Note**
>
> 在接入点策略中授予的权限仅在底层存储桶也允许相同的访问时才有效。您可以通过两种方式实现这一点：
>
> 1. （推荐）将存储桶的访问控制委派给接入点，如将访问控制委派到接入点中所述。
> 2. 将接入点策略中包含的权限添加到底层存储桶的策略中。示例 1 接入点策略示例说明了如何修改底层存储桶策略以允许必要的访问。

Example 1 – 接入点策略授予

以下接入点策略通过账户 *123456789012* 中的接入点 *my-access-point* 向账户 *123456789012* 中的 IAM 用户 *Jane* 授予对具有前缀 *Jane/* 的 GET 和 PUT 对象的权限。

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
```

```
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/
object/Jane/*"
    }]
}
```

Note

要使接入点策略有效地向 *Jane* 授予访问权限，底层存储桶也必须对 *Jane* 允许相同的访问权限。可以将存储桶的访问控制委派到接入点，如将访问控制委派到接入点中所述。或者，您也可以将以下策略添加到底层存储桶中，以便向 Jane 授予必要的权限。请注意，接入点策略和存储桶策略的 Resource 条目不同。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/Jane/*"
    }]
}
```

Example 2 – 带标签条件的接入点策略

以下接入点策略通过账户 *123456789012* 中的接入点 *my-access-point* 向账户 *123456789012* 中的 IAM 用户 *Mateo* 授予对 GET 对象的权限，这些权限具有值设为 *finance* 的标签键 *data*。

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect":"Allow",
```

```
        "Principal" : {
            "AWS": "arn:aws:iam::123456789012:user/Mateo"
        },
        "Action":"s3:GetObject",
        "Resource" : "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/
object/*",
        "Condition" : {
            "StringEquals": {
                "s3:ExistingObjectTag/data": "finance"
            }
        }
    }]
}
```

Example 3 – 允许查看存储桶列示内容的接入点策略

以下接入点策略通过账户 *123456789012* 中的接入点 *my-access-point* 授予账户 *123456789012* 中的 IAM 用户 Arnav 查看底层存储桶中包含的对象的权限。

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Arnav"
        },
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point"
    }]
}
```

Example 4 – 服务控制策略

以下服务控制策略要求使用虚拟私有云（VPC）网络起源创建所有新的接入点。实施此策略时，组织中的用户无法创建可从 Internet 访问的新接入点。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Deny",
        "Action": "s3:CreateAccessPoint",
```

```
        "Resource": "*",
        "Condition": {
            "StringNotEquals": {
                "s3:AccessPointNetworkOrigin": "VPC"
            }
        }
    }]
}
```

Example 5 – 将 S3 操作限制为 VPC 网络起源的存储桶策略

以下存储桶策略限制为只能通过具有 VPC 网络起源的接入点来访问存储桶 *amzn-s3-demo-bucket* 的所有 S3 对象操作。

> **Important**
>
> 在使用类似于此示例所示语句的语句之前，请确保您不需要使用接入点不支持的特征，例如跨区域复制。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:BypassGovernanceRetention",
                "s3:DeleteObject",
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersion",
                "s3:DeleteObjectVersionTagging",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging",
                "s3:ListMultipartUploadParts",
```

```
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectLegalHold",
                "s3:PutObjectRetention",
                "s3:PutObjectTagging",
                "s3:PutObjectVersionAcl",
                "s3:PutObjectVersionTagging",
                "s3:RestoreObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:AccessPointNetworkOrigin": "VPC"
                }
            }
        }
    ]
}
```

条件键

S3 接入点具有条件键，这些键可在 IAM 策略中用来控制对资源的访问。以下条件键仅代表 IAM 策略的一部分。有关完整策略示例，请参阅接入点策略示例、the section called “将访问控制委派到接入点”和the section called “授予跨账户接入点的权限”。

s3:DataAccessPointArn

此示例显示一个您可以用来匹配接入点 ARN 的字符串。以下示例匹配区域 *us-west-2* 中 AWS 账户 *123456789012* 的所有接入点：

```
"Condition" : {
    "StringLike": {
        "s3:DataAccessPointArn": "arn:aws:s3:us-west-2:123456789012:accesspoint/*"
    }
}
```

s3:DataAccessPointAccount

此示例显示一个字符串运算符，您可以使用它匹配接入点拥有者的账户 ID。以下示例匹配 AWS 账户 *123456789012* 拥有的所有接入点。

```
"Condition" : {
```

```
    "StringEquals": {
        "s3:DataAccessPointAccount": "123456789012"
    }
}
```

s3:AccessPointNetworkOrigin

此示例显示一个字符串运算符，您可以使用它来匹配网络起源（Internet 或 VPC）。以下示例仅匹配起源为 VPC 的接入点。

```
"Condition" : {
    "StringEquals": {
        "s3:AccessPointNetworkOrigin": "VPC"
    }
}
```

有关在 Amazon S3 中使用条件键的更多信息，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

将访问控制委派到接入点

您可以将存储桶的访问控制委托给存储桶的接入点。以下示例存储桶策略允许对存储桶拥有者账户所拥有的所有接入点进行完全访问。因此，对此存储桶的所有访问都由附加到其接入点的策略控制。我们建议您以这种方式为所有不需要直接访问存储桶的使用案例配置存储桶。

Example 6 – 将访问控制委派给接入点的存储桶策略

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's account
 ID" }
        }
    }]
}
```

授予跨账户接入点的权限

要为另一个账户拥有的存储桶创建接入点，您必须首先通过指定存储桶名称和账户拥有者 ID 来创建接入点。然后，存储桶拥有者必须更新存储桶策略以授权来自接入点的请求。创建接入点与创建 DNS CNAME 类似，因为接入点不提供对存储桶内容的访问权限。所有存储桶访问权限均由存储桶策略控制。以下示例存储桶策略允许从受信任 AWS 账户拥有的接入点对存储桶进行 GET 和 LIST 请求。

将 *Bucket ARN* 替换为桶的 ARN。

Example 7 – 将权限委托给其他 AWS 账户的存储桶策略

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : ["s3:GetObject","s3:ListBucket"],
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Access point owner's
 account ID" }
        }
    }]
}
```

创建接入点

Amazon S3 提供了创建和管理接入点的功能。您可以使用 AWS Management Console、AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API 创建 S3 接入点。

默认情况下，您可为每个 AWS 账户在每个区域创建最多 10000 个接入点。如果您需要在单个区域为一个账户使用超过 10000 个接入点，则可以请求增加服务限额。有关服务限额和请求增加限额的更多信息，请参阅《AWS 一般参考》中的 AWS 服务限额。

> **Note**
>
> 由于您可能希望公布您的接入点名称，以便其他用户可以使用此接入点，因此请避免在接入点名称中包含敏感信息。接入点名称将在称为域名系统（DNS）的可公开访问的数据库中得以发布。

命名 Amazon S3 接入点的规则

接入点名称必须符合以下条件：

- 在单个 AWS 账户和区域内必须是唯一的
- 必须遵守 DNS 命名限制
- 必须以数字或小写字母开头
- 长度必须介于 3-50 个字符之间
- 不能以连字符（-）开头或结尾
- 不能包含下划线（_）、大写字母或句点（.）
- 不得以后缀 `-s3alias` 结尾。此后缀是为接入点别名预留的。有关更多信息，请参阅 为您的 S3 存储桶接入点使用存储桶式别名。

要创建接入点，请参阅以下主题。

主题

- 创建接入点
- 创建限制到 Virtual Private Cloud 的接入点
- 管理接入点的公有访问

创建接入点

一个接入点只与一个 Amazon S3 存储桶相关联。如果您想使用您的 AWS 账户中的存储桶，则必须首先创建存储桶。有关创建存储桶的更多信息，请参阅创建、配置和使用 Amazon S3 存储桶。

您也可以创建与另一个 AWS 账户中的存储桶关联的跨账户接入点，只要您知道存储桶名称和存储桶拥有者的账户 ID。但是，在获得存储桶拥有者授予的权限之前，创建跨账户接入点不会授予您访问存储桶中数据的权限。存储桶拥有者必须通过存储桶策略向接入点拥有者的账户（您的账户）授予对存储桶的访问权限。有关更多信息，请参阅 授予跨账户接入点的权限。

默认情况下，您可为每个 AWS 账户在每个区域创建最多 10000 个接入点。如果您需要在单个区域为一个账户使用超过 10000 个接入点，则可以请求增加服务限额。有关服务限额和请求增加限额的更多信息，请参阅《AWS 一般参考》中的 AWS 服务限额。

以下示例演示如何使用 AWS CLI 和 S3 控制台创建接入点。有关如何使用 REST API 创建接入点的信息，请参阅《Amazon Simple Storage Service API 参考》中的 CreateAccessPoint。

使用 S3 控制台

创建接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要在其中创建接入点的区域。
3. 在左侧导航窗格中，选择接入点。
4. 在接入点页面上，选择创建接入点。
5. 在接入点名称字段中，输入接入点的名称。有关命名接入点的更多信息，请参阅命名 Amazon S3 接入点的规则。
6. 对于存储桶名称，指定要与接入点结合使用的 S3 存储桶。

   要使用您账户中的存储桶，请选择选择此账户中的存储桶，然后输入或浏览目标存储桶。

   要使用其他 AWS 账户中的存储桶，请选择指定另一个账户中的存储桶，然后输入该存储桶的 AWS 账户 ID 和名称。

   > **Note**
   >
   > 如果您使用其他 AWS 账户中的存储桶，存储桶拥有者必须更新存储桶策略才能授权来自接入点的请求。有关存储桶策略示例，请参阅 授予跨账户接入点的权限。

7. 选择网络来源。如果您选择 Virtual private cloud（VPC），请输入要与接入点一起使用的 VPC ID。

   有关接入点的网络源的更多信息，请参阅创建限制到 Virtual Private Cloud 的接入点。

8. 在 Block Public Access settings for this Access Point（阻止此接入点的公有访问设置）下，选择要应用于接入点的屏蔽公共访问权限设置。默认情况下，为新的接入点启用所有屏蔽公共访问权限设置。除非您有特定的需求要禁用任何一个设置，建议您将所有设置保持为启用状态。

   > **Note**
   >
   > 创建接入点后，无法更改其屏蔽公共访问权限设置。

有关将 Amazon S3 屏蔽公共访问权限与接入点结合使用的更多信息，请参阅管理接入点的公有访问。

9. （可选）在 Access point policy（接入点策略）- optional（可选）下，指定接入点策略。在保存策略之前，确保解决任何安全警告、错误、一般警告和建议。有关指定接入点策略的更多信息，请参阅接入点策略示例。
10. 选择 Create access point（创建接入点）。

使用 AWS CLI

以下示例命令为账户 *111122223333* 中的存储桶 *amzn-s3-demo-bucket* 创建了一个名为 *example-ap* 的接入点。要创建接入点，您需要向 Amazon S3 发送一个指定以下内容的请求：

- 接入点名称。有关命名规则的信息，请参阅 the section called “命名 Amazon S3 接入点的规则”。
- 要与接入点关联的存储桶的名称。
- 拥有存储桶的 AWS 账户的账户 ID。

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket amzn-s3-demo-bucket
```

当您使用其他 AWS 账户中的存储桶创建接入点时，请包括 `--bucket-account-id` 参数。以下示例命令使用位于 AWS 账户 *444455556666* 中的存储桶 *amzn-s3-demo-bucket2* 在 AWS 账户 *111122223333* 中创建接入点。

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket amzn-s3-demo-bucket --bucket-account-id 444455556666
```

创建限制到 Virtual Private Cloud 的接入点

创建接入点时，您可以选择让接入点可从 Internet 访问，也可以指定通过该接入点发出的所有请求都必须来自特定 Virtual Private Cloud（VPC）。可从 Internet 访问的接入点被认为是具有 `Internet` 网络起源。它可以在 Internet 上的任何位置使用，但受到接入点、底层存储桶和相关资源（如请求的对象）的任何其他访问限制的约束。只能从指定 VPC 访问的接入点的网络起源为 VPC，并且 Amazon S3 会拒绝从该 VPC 之外的来源向接入点发出的任何请求。

> **Important**
>
> 您只能在创建接入点时指定接入点的网络起源。创建接入点后，无法更改其网络起源。

要将接入点限制为仅限于 VPC 访问，请在创建接入点的请求中包括 `VpcConfiguration` 参数。在 `VpcConfiguration` 参数中，您指定希望能够使用接入点的 VPC ID。如果请求是通过接入点发出的，则该请求必须源自 VPC，否则 Amazon S3 将拒绝该请求。

您可以使用 AWS CLI、AWS SDK 或 REST API 检索接入点的网络起源。如果接入点指定了 VPC 配置，则其网络起源为 `VPC`。否则，接入点的网络起源为 `Internet`。

Example

示例：创建仅限 VPC 访问的接入点

以下示例为账户 `123456789012` 中的存储桶 `example-bucket` 创建一个名为 `example-vpc-ap` 的接入点，该接入点仅允许从 VPC `vpc-1a2b3c` 访问。然后，该示例验证新接入点是否具有 `VPC` 网络起源。

AWS CLI

```
aws s3control create-access-point --name example-vpc-ap --account-id 123456789012 --
bucket example-bucket --vpc-configuration VpcId=vpc-1a2b3c
```

```
aws s3control get-access-point --name example-vpc-ap --account-id 123456789012

{
    "Name": "example-vpc-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "VPC",
    "VpcConfiguration": {
        "VpcId": "vpc-1a2b3c"
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
```

```
}
```

要将接入点与 VPC 搭配使用，您必须修改 VPC 端点的访问策略。VPC 端点允许流量从您的 VPC 流向 Amazon S3。它们具有访问控制策略，用于控制如何允许 VPC 内的资源与 Amazon S3 交互。仅当 VPC 端点策略同时授予对接入点和底层存储桶的访问权限时，从 VPC 通过接入点到 S3 的请求才会成功。

> **Note**
>
> 要使资源只能在 VPC 中访问，请确保为 VPC 端点创建私有托管区域。要使用私有托管区域，请修改您的 VPC 设置，以便 VPC 网络属性 `enableDnsHostnames` 和 `enableDnsSupport` 设置为 `true`。

以下示例策略语句配置 VPC 端点，以允许对名为 `awsexamplebucket1` 的存储桶和名为 `example-vpc-ap` 的接入点执行 `GetObject` 调用。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*",
            "arn:aws:s3:us-west-2:123456789012:accesspoint/example-vpc-ap/object/*"
        ]
    }]
}
```

> **Note**
>
> 此示例中的 "Resource" 声明使用 Amazon 资源名称（ARN）指定接入点。有关接入点 ARN 的更多信息，请参阅 使用接入点。

有关 VPC 端点策略的更多信息，请参阅《VPC 用户指南》中的使用 Amazon S3 的端点策略。

管理接入点的公有访问

Amazon S3 接入点支持每个接入点的独立屏蔽公共访问权限 设置。创建接入点时，可以指定应用于该接入点的屏蔽公共访问权限设置。对于通过接入点发出的任何请求，Amazon S3 会评估该接入点、底层存储桶和存储桶拥有者账户的屏蔽公共访问权限设置。如果这些设置中的任何一个指示应阻止请求，则 Amazon S3 会拒绝请求。

有关 S3 屏蔽公共访问权限特征的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

> **Important**
>
> - 默认情况下为接入点启用所有屏蔽公共访问权限设置。您必须明确禁用不希望应用于接入点的任何设置。
> - Amazon S3 当前不支持在创建接入点之后更改接入点的屏蔽公共访问权限设置。

Example

示例：创建具有自定义屏蔽公共访问权限设置的接入点

此示例为账户 123456789012 中的存储桶 example-bucket 创建名为 example-ap 且具有非默认屏蔽公共访问权限设置的接入点。然后，该示例检索新接入点的配置，以验证其“屏蔽公共访问权限”设置。

AWS CLI

```
aws s3control create-access-point --name example-ap --account-id
 123456789012 --bucket example-bucket --public-access-block-configuration
 BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=true,RestrictPublicBuckets=t
```

```
aws s3control get-access-point --name example-ap --account-id 123456789012

{
    "Name": "example-ap",
    "Bucket": "example-bucket",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": false,
```

```
        "IgnorePublicAcls": false,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

使用接入点

您可以使用 AWS Management Console、AWS CLI、AWS SDK 或 S3 REST API 通过接入点来访问 Amazon S3 存储桶中的对象。

接入点具有 Amazon Resource Name（ARN）。接入点 ARN 与存储桶 ARN 类似，但它们是明确键入的，用于对接入点的区域和接入点的拥有者的 AWS 账户 ID 进行编码。有关 ARN 的更多信息，请参阅《AWS 一般参考》中的 Amazon 资源名称（ARN）。

接入点 ARN 使用以下格式：`arn:aws:s3:region:account-id:accesspoint/resource`。例如：

- arn:aws:s3:us-west-2:123456789012:accesspoint/test 表示由区域 `test` 中的账户 `123456789012` 拥有的名为 `us-west-2` 的接入点。
- arn:aws:s3:us-west-2:123456789012:accesspoint/* 表示区域 `123456789012` 中的账户 `us-west-2` 拥有的所有接入点。

通过接入点访问的对象的 ARN 使用以下格式：`arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource`。例如：

- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01 表示通过区域 `unit-01` 中的账户 `test` 拥有的名为 `123456789012` 的接入点访问的对象 `us-west-2`。
- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/* 表示区域 `test` 的账户 `123456789012` 中的接入点 `us-west-2` 的所有对象。
- arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01/finance/* 表示区域 `unit-01/finance/` 的账户 `test` 中的接入点 `123456789012` 的所有前缀为 `us-west-2` 的对象。

通过 S3 访问点访问存储桶

S3 访问点仅支持虚拟主机式寻址。要通过访问点对存储桶进行寻址，请使用以下格式。

```
https://AccessPointName-AccountId.s3-accesspoint.region.amazonaws.com
```

> **Note**
>
> - 如果您的访问点名称包含破折号 (-) 字符，请在 URL 中包含破折号，然后在账户 ID 之前插入另一个破折号。例如，要在区域 finance-docs 中使用由账户 123456789012 拥有的名为 us-west-2 访问点 ，则相应的 URL 为 https://finance-docs-123456789012.s3-accesspoint.us-west-2.amazonaws.com。
> - S3 访问点不支持通过 HTTP 进行访问，只支持通过 HTTPS 进行安全访问。

主题

如果您有虚拟私有云（VPC），请参阅使用 VPC 端点和 S3 接入点管理 Amazon S3 访问权限。

监控和记录接入点

Amazon S3 记录通过接入点发出的请求以及对管理接入点（如 CreateAccessPoint 和 GetAccessPointPolicy）的 API 发出的请求。如需监控和管理使用模式，您还可以为接入点配置 Amazon CloudWatch Logs 请求指标。

主题

CloudWatch 请求指标

如需了解和提高使用接入点的应用程序的性能，您可以使用 Amazon S3 的 CloudWatch 请求指标。请求指标帮助您监控 Amazon S3 请求，以快速识别运行问题，并对问题采取措施。

默认情况下，请求指标在存储桶级别可用。但是，您可以使用共享前缀、对象标签或接入点为请求指标定义筛选器。当您创建接入点筛选器时，请求指标配置包括了对指定接入点的请求。您可以接收指标、设置警报和访问控制面板，以查看通过此接入点执行的实时操作。

您必须通过在控制台中配置请求指标或使用 Amazon S3 API 来选择使用请求指标。请求指标在一定的处理延迟之后每隔一分钟提供一次。请求指标的费率与CloudWatch 自定义指标的费率相同。有关更多信息，请参阅 Amazon CloudWatch 定价。

如需创建按接入点筛选的请求指标配置，请参阅 创建按前缀，对象标签或接入点筛选的指标配置。

请求日志

您可以通过使用服务器访问日志和 AWS CloudTrail，记录通过接入点发出的请求和对管理接入点（如 `CreateAccessPoint` 和 `GetAccessPointPolicy`,）的 API 发出的请求。

通过接入点发出的请求 CloudTrail 日志条目将在日志的 `resources` 部分包含接入点 ARN。

例如，假设您有以下配置：

- 区域 `us-west-2` 中名为 `amzn-s3-demo-bucket1` 的存储桶包含名为 `my-image.jpg` 的对象。
- 与 `amzn-s3-demo-bucket1` 关联且名为 `my-bucket-ap` 的接入点
- `123456789012` 的 AWS 账户 ID

以下示例显示了上述配置的CloudTrail 日志条目的 `resources` 部分：

```
"resources": [
        {"type": "AWS::S3::Object",
            "ARN": "arn:aws:s3:::amzn-s3-demo-bucket1/my-image.jpg"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::Bucket",
            "ARN": "arn:aws:s3:::amzn-s3-demo-bucket1"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::AccessPoint",
            "ARN": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-bucket-ap"
        }
    ]
```

有关 S3 服务器访问日志的更多信息，请参阅使用服务器访问日志记录来记录请求。有关 AWS CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》中的什么是 AWS CloudTrail？。

通过 Amazon S3 控制台使用 Amazon S3 接入点

本部分介绍如何通过 AWS Management Console来管理和使用 Amazon S3 接入点。在开始之前，请导航至要管理或使用的接入点的详细信息页面，如以下程序中所述。

主题

列出账户的接入点

列出在 AWS 账户中创建的所有接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要列出其接入点的区域。
3. 在控制台左侧的导航窗格中，选择 Access points（接入点）。
4. 在接入点页面的接入点下，查看您的 AWS 区域中的接入点。
5. （可选）通过在“Region（区域）”下拉菜单旁边的文本字段中输入名称，按名称搜索接入点。
6. 选择要管理或使用的接入点的名称。

列出存储桶的接入点

列出您的 AWS 账户中单个存储桶的所有接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前显示的 AWS 区域的名称，然后选择要列出其接入点的区域。
3. 在控制台左侧的导航窗格中，选择 Buckets（存储桶）。
4. 在 Buckets（存储桶）页面上，选择要列出其接入点的存储桶的名称。
5. 在存储桶详细信息页面上，选择 Access points（接入点）选项卡。
6. 选择要管理或使用的接入点的名称。

查看接入点的配置详细信息

1. 导航到要查看其详细信息的接入点的详细信息页面，如列出账户的接入点中所述。
2. 在 Access point overview（接入点概述）下，查看所选接入点的配置详细信息和属性。

使用接入点

1. 导航到要使用的接入点的详细信息页面，如列出账户的接入点中所述。
2. 在 Objects（对象）选项卡下，选择要通过接入点访问的一个或多个对象的名称。在对象操作页面上，控制台在存储桶名称上方显示一个标签，该标签显示您当前正在使用的接入点。在使用接入点时，您只能执行接入点权限允许的对象操作。

> **Note**
>
> - 控制台视图始终显示存储桶中的所有对象。如本过程中所述使用接入点会限制您可以对这些对象执行的操作，但不会限制您是否能够看到这些对象存在于存储桶中。
> - S3 管理控制台不支持使用 Virtual Private Cloud（VPC）接入点访问存储桶资源。要从 VPC 接入点访问存储桶资源，请使用 AWS CLI、AWS SDK 或 Amazon S3 REST API。

查看接入点的屏蔽公共访问权限设置

1. 导航到要查看其设置的接入点的详细信息页面，如列出账户的接入点中所述。
2. 选择 Permissions。
3. 在 Access point policy（接入点策略）下，查看接入点的屏蔽公共访问权限设置。

> **Note**
>
> 创建接入点后，您无法更改接入点的屏蔽公共访问权限设置。

编辑接入点策略

1. 导航到要编辑其策略的接入点的详细信息页面，如列出账户的接入点中所述。
2. 选择 Permissions。
3. 在 Access point policy（接入点策略）下，选择 Edit（编辑）。
4. 在文本字段中输入接入点策略。控制台会自动显示接入点的 Amazon Resource Name（ARN），您可以在策略中使用该名称。

删除接入点

1. 导航到您的账户或特定存储桶的接入点列表，如列出账户的接入点中所述。
2. 选择要删除的接入点名称旁边的选项按钮。
3. 选择 Delete。
4. 在显示的文本字段中输入接入点名称，然后选择 Delete（删除），确认您要删除接入点。

为您的 S3 存储桶接入点使用存储桶式别名

当您创建接入点时，Amazon S3 会自动生成一个别名。您可以使用该别名来替代 Amazon S3 存储桶名称以进行数据访问。对于接入点数据面板操作，您可以使用此接入点别名，而不是 Amazon 资源名称（ARN）。有关这些操作的列表，请参阅接入点与AWS服务的兼容性。

下面显示了名为 *my-access-point* 的接入点的示例 ARN 和接入点别名。

- ARN – `arn:aws:s3:`*region*`:`*account-id*`:accesspoint/`*my-access-point*
- 接入点别名 – *my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b*`-s3alias`

有关 ARN 的更多信息，请参阅《AWS 一般参考》中的 Amazon 资源名称（ARN）。

接入点别名

接入点别名是在与 Amazon S3 存储桶相同的命名空间中创建的。此别名自动生成，无法更改。接入点别名符合有效 Amazon S3 存储桶名称的所有要求，包括以下部分：

access point prefix-*metadata*-s3alias

> **Note**
>
> -s3alias 后缀是为接入点别名预留的，不能用于存储桶或接入点名称。有关 Amazon S3 存储桶命名规则的更多信息，请参阅存储桶命名规则。

接入点别名使用案例和限制条件

采用接入点时，您可以使用接入点别名，而无需对代码进行大量更改。

当您创建接入点时，Amazon S3 会自动生成一个接入点别名，如下例所示。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-access-point --bucket amzn-s3-demo-bucket1 --name my-access-point
 --account-id 111122223333
{
    "AccessPointArn":
    "arn:aws:s3:region:111122223333:accesspoint/my-access-point",
    "Alias": "my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-s3alias"
}
```

在任何数据层面操作中，您都可以使用此接入点别名而不是 Amazon S3 存储桶名称。有关这些操作的列表，请参阅接入点与AWS服务的兼容性。

以下有关 get-object 命令的 AWS CLI 示例使用存储桶的接入点别名来返回有关指定对象的信息。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3api get-object --bucket my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-
s3alias --key dir/my_data.rtf my_data.rtf

{
```

```
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
}
```

限制

- 客户无法配置别名。
- 无法在接入点上删除、修改或禁用别名。
- 在某些数据面板操作中，您可以使用此接入点别名，而不是 Amazon S3 存储桶名称。有关这些操作的列表，请参阅接入点与 S3 操作的兼容性。
- 您不能将一个接入点别名用于多个 Amazon S3 控制层面操作。有关 Amazon S3 控制层面操作的列表，请参阅《Amazon Simple Storage Service API 参考》中的 Amazon S3 控制。
- 您不能使用 S3 接入点别名作为 Amazon S3 控制台中移动操作的源或目标。
- 别名不能用于 AWS Identity and Access Management（IAM）策略中。
- 别名不能用作 S3 服务器访问日志的日志记录目标。
- 别名不能用作 AWS CloudTrail 日志的日志记录目标。
- Amazon SageMaker GroundTruth 不支持接入点别名。

将接入点与兼容的 Amazon S3 操作结合使用

以下示例演示如何在 Amazon S3 中将接入点与兼容操作结合使用。

主题

- 接入点与AWS服务的兼容性
- 接入点与 S3 操作的兼容性
- 通过接入点请求对象
- 通过接入点别名上传对象
- 通过接入点删除对象
- 通过接入点别名列出对象
- 通过接入点向对象添加标签集

- 使用 ACL 通过接入点授予访问权限

接入点与AWS服务的兼容性

Amazon S3 接入点别名允许需要 S3 存储桶名称的应用程序轻松使用接入点。在使用 S3 存储桶名称访问 S3 中数据的位置，您可以使用 S3 接入点别名。有关更多信息，请参阅 接入点别名使用案例和限制条件。

接入点与 S3 操作的兼容性

您可以使用 Amazon S3 API 的以下子集通过接入点来访问存储桶：下列所有操作都可以接受接入点 ARN 或接入点别名：

S3 操作

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject（仅限同区域副本）
- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetBucketAcl
- GetBucketCors
- GetBucketLocation
- GetBucketNotificationConfiguration
- GetBucketPolicy
- GetObject
- GetObjectAcl
- GetObjectAttributes
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadBucket
- HeadObject

- ListMultipartUploads
- ListObjects
- ListObjectsV2
- ListObjectVersions
- ListParts
- Presign
- PutObject
- PutObjectLegalHold
- PutObjectRetention
- PutObjectAcl
- PutObjectTagging
- RestoreObject
- UploadPart
- UploadPartCopy（仅限同区域副本）

通过接入点请求对象

以下示例通过区域 us-west-2 中的账户 ID 123456789012 拥有的接入点 prod 来请求对象 my-image.jpg，并将下载的文件另存为 download.jpg。

AWS CLI

```
aws s3api get-object --key my-image.jpg --bucket arn:aws:s3:us-
west-2:123456789012:accesspoint/prod download.jpg
```

通过接入点别名上传对象

以下示例通过区域 us-west-2 中的账户 ID 123456789012 拥有的接入点别名 my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias 来上传对象 my-image.jpg。

AWS CLI

```
aws s3api put-object --bucket my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-
s3alias --key my-image.jpg --body my-image.jpg
```

通过接入点删除对象

以下示例通过区域 us-west-2 中的账户 ID 123456789012 拥有的接入点 prod 来删除对象 my-image.jpg。

AWS CLI

```
aws s3api delete-object --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod
 --key my-image.jpg
```

通过接入点别名列出对象

以下示例通过区域 us-west-2 中的账户 ID 123456789012 拥有的接入点别名 my-access-point-hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias 来列出对象。

AWS CLI

```
aws s3api list-objects-v2 --bucket my-access-point-
hrzrlukc5m36ft7okagglf3gmwluquse1b-s3alias
```

通过接入点向对象添加标签集

以下示例通过区域 us-west-2 中的账户 ID 123456789012 拥有的接入点 prod 向现有对象 my-image.jpg 添加标签集。

AWS CLI

```
aws s3api put-object-tagging --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/
prod --key my-image.jpg --tagging TagSet=[{Key="finance",Value="true"}]
```

使用 ACL 通过接入点授予访问权限

以下示例通过区域 us-west-2 中的账户 ID 123456789012 拥有的接入点 prod 向现有对象 my-image.jpg 应用 ACL。

AWS CLI

```
aws s3api put-object-acl --bucket arn:aws:s3:us-west-2:123456789012:accesspoint/prod
 --key my-image.jpg --acl private
```

接入点限制和局限性

Amazon S3 接入点具有以下限制和局限性：

- 每个接入点只与一个存储桶相关联，您必须在创建接入点时指定该存储桶。创建接入点后，您无法将它与另一个存储桶关联。但是，您可以删除接入点，然后创建具有相同名称的另一个接入点并将新的接入点与另一个存储桶关联。
- 接入点名称必须满足某些条件。有关命名接入点的更多信息，请参阅命名 Amazon S3 接入点的规则。
- 创建接入点后，无法更改其 Virtual Private Cloud（VPC）配置。
- 接入点策略的大小限制为 20 KB。
- 每个区域每个 AWS 账户最多可以创建 10000 个接入点。如果您需要在单个区域为一个账户使用超过 10000 个接入点，则可以请求增加服务限额。有关服务限额和请求增加限额的更多信息，请参阅《AWS 一般参考》中的 AWS 服务限额。
- 在拥有超过 1000 个接入点的 AWS 区域中，您无法在 Amazon S3 控制台中按名称搜索接入点。
- 您不能将接入点用作 S3 复制的目标。有关复制的更多信息，请参阅 复制对象概述。
- 您不能使用 S3 接入点别名作为 Amazon S3 控制台中移动操作的源或目标。
- 您只能使用虚拟主机式 URL 为接入点寻址。有关虚拟主机式寻址的更多信息，请参阅访问和列出 Amazon S3 存储桶。
- 控制接入点功能（例如，`PutAccessPoint` 和 `GetAccessPointPolicy`）的 API 操作不支持跨账户调用。
- 使用 REST API 向接入点发出请求时，必须使用 AWS 签名版本 4。有关对请求进行身份验证的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的验证请求（AWS 签名版本 4）。
- 接入点仅支持通过 HTTPS 发出请求。对于通过 HTTP 发出的任何请求，Amazon S3 都会自动以 HTTP 重定向作为响应，从而将请求升级到 HTTPS。
- 接入点不支持匿名访问。
- 除非从存储桶拥有者向您授予权限，否则跨账户接入点不会授予您对数据的访问权限。存储桶拥有者始终保留对数据访问的最终控制权，并且必须更新存储桶策略才能授权来自跨账户接入点的请求。要查看存储桶策略示例，请参阅配置使用接入点的 IAM 策略。

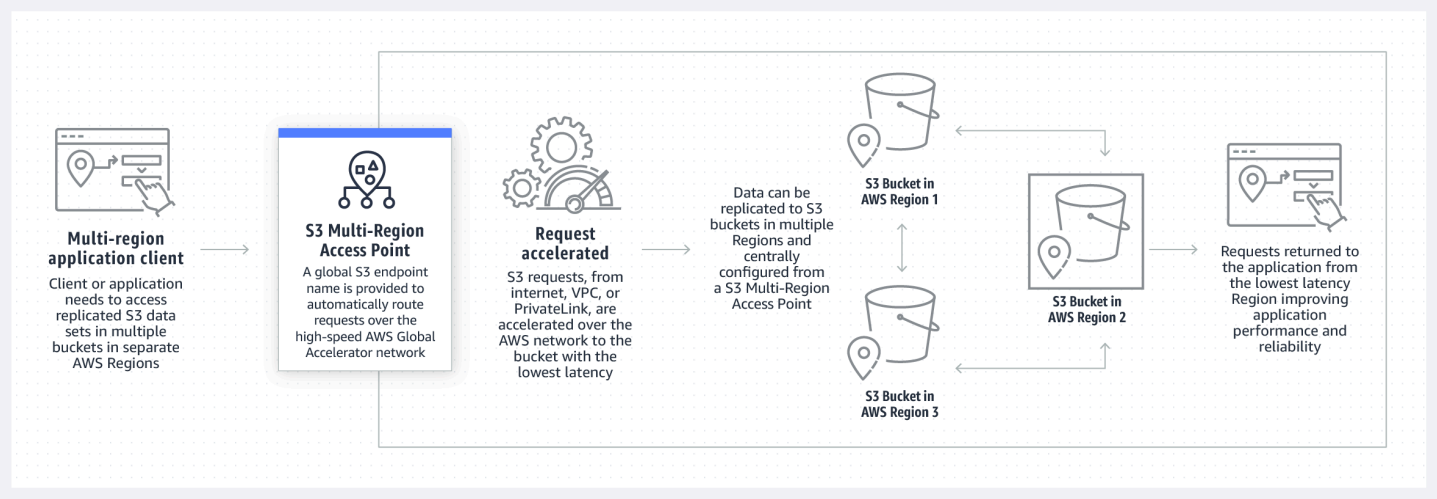
- 当您在 Amazon S3 控制台中查看跨账户接入点时，访问列显示未知。Amazon S3 控制台无法确定是否已授予针对关联存储桶和对象的公共访问权限。除非您需要针对特定使用案例进行公共配置，否则我们建议您和存储桶拥有者屏蔽对接入点和存储桶的所有公共访问权限。有关更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

Amazon S3 中的多区域接入点

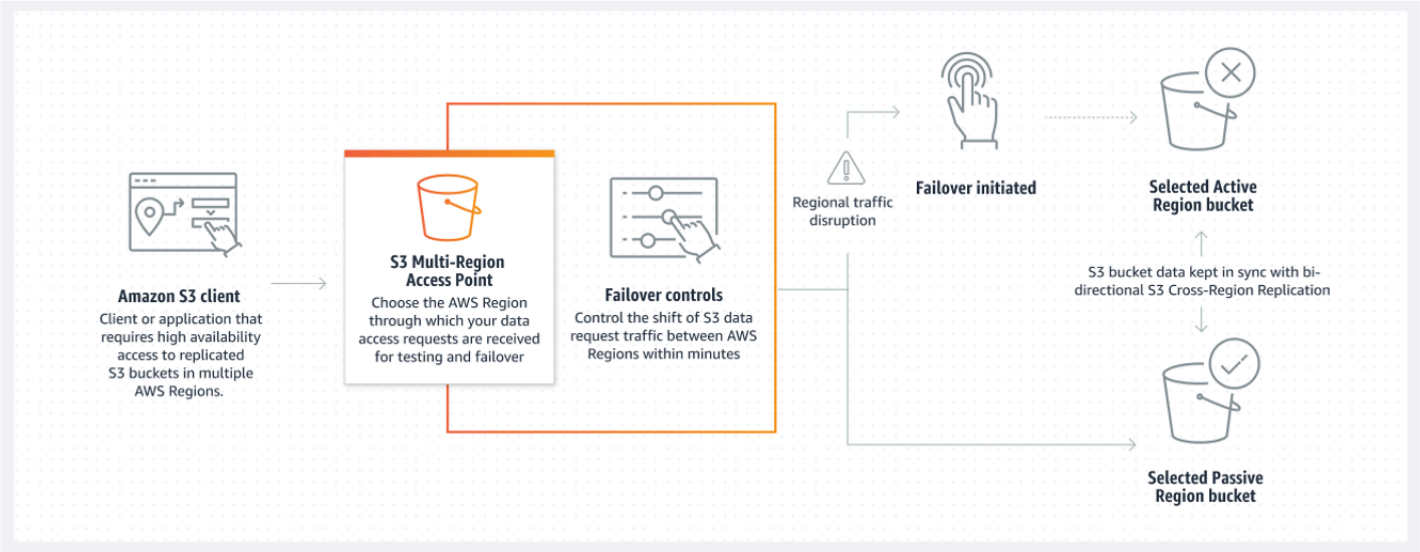
Amazon S3 多区域接入点提供了一个全局端点，应用程序可以使用该端点来满足来自位于多个 AWS 区域中的 S3 桶的请求。您可以使用多区域接入点通过单个区域中使用的相同架构构建多区域应用程序，然后在世界任何地方运行这些应用程序。多区域接入点不是通过拥挤的公共互联网发送请求，而是提供内置的网络恢复能力，加速向 Amazon S3 发送基于互联网的请求。向多区域接入点全局端点发出的应用程序请求使用 AWS Global Accelerator，以自动通过 AWS 全局网络路由到具有活动的路由状态且距离最近的 S3 桶。

创建多区域接入点时，需要指定一组 AWS 区域，您希望在其中存储要通过该多区域接入点提供的数据。您可以使用 S3 跨区域复制（CRR）在这些区域中的桶之间同步数据。然后，您可以通过多区域接入点全局端点请求或写入数据。Amazon S3 自动为来自距离最近的可用区域的复制数据集提供请求。多区域接入点还与 Amazon 虚拟私有云（VPC）中运行的应用程序兼容，包括使用AWS PrivateLink：表示 Amazon S3的应用程序。

下图是处于活动-活动配置中的 Amazon S3 多区域接入点的图形表示。该图显示了 Amazon S3 请求如何自动路由到距离最近的活动 AWS 区域中的桶。



下图是处于活动-被动配置中的 Amazon S3 多区域接入点的图形表示。该图显示如何控制 Amazon S3 数据访问流量以在主动和被动 AWS 区域之间进行失效转移。



要了解有关如何使用多区域接入点的更多信息，请参阅教程：Amazon S3 多区域接入点入门。

主题

- 创建多区域接入点
- 配置多区域接入点以便与 AWS PrivateLink 共同使用
- 通过多区域接入点发出请求

创建多区域接入点

要在 Amazon S3 中创建多区域接入点，请执行以下操作：

- 指定多区域接入点的名称。
- 在每个 AWS 区域中选择一个桶，以便为针对多区域接入点的请求提供服务。
- 为多区域接入点配置 Amazon S3 屏蔽公共访问权限设置。

您可以在创建请求中提供所有这些信息，Amazon S3 将异步处理该请求。Amazon S3 为您提供了一个令牌，您可以使用该令牌监控异步创建请求状态。

确保保存策略之前解决来自 AWS Identity and Access Management Access Analyzer 的安全警告、错误、一般警告和建议。IAM Access Analyzer 将根据 IAM 策略语法和最佳实践运行策略检查，以验证您的策略。这些检查项生成结果并提供可操作的建议，可帮助您编写可操作且符合安全最佳实践的策略。要了解有关使用 IAM Access Analyzer 验证策略的更多信息，请参阅 IAM 用户指南中的 IAM

Access Analyzer 策略验证。要查看 IAM Access Analyzer 返回的警告、错误和建议的列表，请参阅 IAM Access Analyzer 策略检查引用。

使用 API 时，创建多区域接入点的请求是异步的。当提交创建多区域接入点的请求时，Amazon S3 会同步授权该请求。然后，它会立即返回一个您可以用来跟踪创建请求进度的令牌。有关跟踪异步请求以创建和管理多区域接入点的详细信息，请参阅 将多区域接入点与支持的 API 操作结合使用。

创建多区域接入点后，可以为其创建访问控制策略。每个多区域接入点都可以有一个关联的策略。多区域接入点策略是一个基于资源的策略，您可以使用它来按资源、用户或其他条件限制多区域接入点的使用。

> **Note**
>
> 要使应用程序或用户能够通过多区域接入点访问对象，以下两个策略都必须允许此请求：
>
> - 多区域接入点的访问策略
> - 包含对象的底层桶的访问策略
>
> 这两个策略不同时，限制性较强的策略优先。
> 要简化多区域接入点的权限管理，您可以将桶中的访问控制委托给多区域接入点。有关更多信息，请参阅 the section called “多区域接入点策略示例”。

通过现有桶名称或 Amazon 资源名称（ARN）访问桶时，将桶与多区域接入点结合使用不会更改桶的行为。针对桶的所有现有操作将继续像以前一样运行。您在多区域接入点策略中包括的限制仅适用于通过多区域接入点发出的请求。

您可以在创建多区域接入点后更新策略，但无法删除该策略。但是，您可以更新多区域接入点策略以拒绝所有权限。

主题

- 命名 Amazon S3 多区域接入点的规则
- 为 Amazon S3 多区域接入点选择桶的规则
- 创建 Amazon S3 多区域接入点
- 用 Amazon S3 多区域接入点阻止公有访问
- 查看 Amazon S3 多区域接入点配置详细信息
- 删除多区域接入点

命名 Amazon S3 多区域接入点的规则

创建多区域接入点时，您可以为其指定一个名称，该名称是您选择的字符串。创建多区域接入点后，无法更改该接入点的名称。名称在您的 AWS 账户 中必须唯一，必须符合 多区域接入点限制和限制 中列出的命名要求。要帮助您识别多区域接入点，请使用对您、您的组织有意义或能够反映场景的名称。

在调用多区域接入点管理操作，如 `GetMultiRegionAccessPoint` 和 `PutMultiRegionAccessPointPolicy`，您使用此名称。该名称不用于向多区域接入点发送请求，也不需要向使用多区域接入点发出请求的客户端公开。

Amazon S3 创建多区域接入点时，会自动为其分配一个别名。此别名是唯一结尾为 `.mrap` 的字母数字字符串。别名用于构建多区域接入点的主机名和 Amazon Resource Name（ARN）。完全限定名称还基于多区域接入点的别名。

您无法根据其别名确定多区域接入点的名称，因此您可以泄露别名，而不会暴露多区域接入点的名称、用途或所有者。Amazon S3 会为每个新的多区域接入点选择别名，不能更改别名。有关多区域接入点寻址的更多信息，请参阅 通过多区域接入点发出请求。

多区域接入点的别名始终是唯一的，不基于多区域接入点的名称或配置。如果创建一个多区域接入点，然后将其删除并创建另一个具有相同名称和配置的接入点，第二个多区域接入点将具有与第一个不同的别名。新的多区域接入点不得具有与之前的多区域接入点相同的别名。

为 Amazon S3 多区域接入点选择桶的规则

每个多区域接入点都与您要满足请求的区域相关联。多区域接入点必须与每个区域中的一个桶相关联。您可以在请求中指定每个桶的名称来创建多区域接入点。支持多区域接入点的桶可以位于拥有多区域接入点的同一个 AWS 账户中，也可以位于其他 AWS 账户中。

多个多区域接入点可以使用同一个桶。

> **Important**
>
> - 您只能在创建多区域接入点时指定与多区域接入点关联的桶。创建该桶后，您无法从多区域接入点配置中添加、修改或删除桶。要更改桶，您必须删除整个多区域接入点，然后创建新的接入点。
> - 您无法删除属于多区域接入点的桶。如果要删除附加到多区域接入点的桶，请先删除多区域接入点。

- 如果您将其他账户拥有的桶添加到多区域接入点，桶拥有者还必须更新其桶策略，以授予对多区域接入点的访问权限。否则，多区域接入点将无法从该桶检索数据。有关显示如何授予此类访问权限的示例策略，请参阅多区域接入点策略示例。
- 并非所有区域都支持多区域接入点。要查看支持的区域列表，请参阅 多区域接入点限制和限制。

您可以创建复制规则在桶之间同步数据。这些规则让您能够自动将数据从源桶复制到目标桶。将桶连接到多区域接入点不会影响复制的工作方式。后面的部分将介绍如何使用多区域接入点配置复制。

Important

当向多区域接入点发出请求时，多区域接入点不了解多区域接入点中桶的数据内容。因此，收到请求的桶可能不包含所请求的数据。要在 Amazon S3 桶中创建与多区域接入点相关联的一致数据集，我们建议您配置 S3 跨区域复制（CRR）。有关更多信息，请参阅 配置用于多区域接入点的复制。

创建 Amazon S3 多区域接入点

以下示例演示了如何使用 Amazon S3 控制台创建多区域接入点。

使用 S3 控制台

创建多区域接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
3. 选择创建多区域接入点，以开始创建多区域接入点。
4. 在多区域接入点页面上，在多区域接入点名称字段中为多区域接入点提供名称。
5. 选择将与此多区域接入点关联的桶。您可以选择您账户中的桶，也可以从其他账户中选择桶。

> **Note**
>
> 您必须从您的账户或其他账户中添加至少一个桶。另请注意，多区域接入点对于每个 AWS 区域仅支持一个桶。因此，您无法添加来自同一个区域的两个桶。不支持原定设置情况下禁用的 AWS 区域。

- 要添加位于您的账户中的桶，请选择添加桶。将显示您的账户中所有桶的列表。您可以按名称搜索桶，也可以按字母顺序对桶名称进行排序。
- 要从其他账户添加桶，请选择从其他账户添加桶。请确保您知道确切的桶名称和 AWS 账户 ID，因为您无法搜索或浏览其他账户中的桶。

> **Note**
>
> 您必须输入有效的 AWS 账户 ID 和桶名称。桶还必须位于支持的区域中，否则在尝试创建多区域接入点时会遇到错误。有关支持多区域接入点的区域列表，请参阅多区域接入点限制和局限性。

6. （可选）如果您需要移除已添加的桶，请选择移除。

> **Note**
>
> 创建完此多区域接入点后，您无法在该多区域接入点中添加或移除桶。

7. 在阻止接入点的公有访问设置下，选择要应用于多区域接入点的阻止公有访问设置。默认情况下为新的多区域接入点启用所有阻止公有访问设置。除非您有特定的需求要禁用任何一个设置，我们建议您保持启用所有设置。

> **Note**
>
> 创建多区域接入点后，您无法更改多区域接入点的屏蔽公共访问权限设置。因此，如果您要阻止公有访问，请在创建多区域接入点之前，确保您的应用程序在没有公有访问的情况下正常运行。

8. 选择创建多区域接入点。

> **Important**
>
> 当您将其他账户拥有的桶添加到多区域接入点时，桶拥有者还必须更新其桶策略，以授予对多区域接入点的访问权限。否则，多区域接入点将无法从该桶检索数据。有关显示如何授予此类访问权限的示例策略，请参阅多区域接入点策略示例。

使用 AWS CLI

您可以使用 AWS CLI 创建多区域接入点。当您创建多区域接入点时，您必须提供它将支持的所有桶。创建多区域接入点后，无法将桶添加到该多区域接入点。

以下示例使用 AWS CLI 创建一个具有两个桶的多区域接入点。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-multi-region-access-point --account-id 111122223333 --details '{
        "Name": "simple-multiregionaccesspoint-with-two-regions",
        "PublicAccessBlock": {
            "BlockPublicAcls": true,
            "IgnorePublicAcls": true,
            "BlockPublicPolicy": true,
            "RestrictPublicBuckets": true
        },
        "Regions": [
            { "Bucket": "amzn-s3-demo-bucket1" },
            { "Bucket": "amzn-s3-demo-bucket2" }
        ]
    }' --region us-west-2
```

用 Amazon S3 多区域接入点阻止公有访问

每个多区域接入点都有不同的 Amazon S3 阻止公有访问设置。这些设置与拥有多区域接入点和底层桶的 AWS 账户的屏蔽公共访问权限设置一起运行。

当 Amazon S3 授权请求时，会应用这些设置的最严格组合。如果任何这些资源（多区域接入点拥有者账户、底层桶或桶拥有者账户）的屏蔽公共访问权限设置屏蔽对所请求的操作或资源的访问，Amazon S3 将拒绝该请求。

除非您特别需要禁用其中任何设置，我们建议您启用所有阻止公共访问设置。默认情况下，为多区域接入点启用所有阻止公有访问设置。如果启用了屏蔽公共访问权限，则多区域接入点将无法接受基于互联网的请求。

> **Important**
>
> 在创建多区域接入点之后，您无法更改其屏蔽公共访问权限设置。

有关 Amazon S3 阻止公有访问的更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

查看 Amazon S3 多区域接入点配置详细信息

以下示例演示如何使用 Amazon S3 控制台查看多区域接入点的配置详细信息。

使用 S3 控制台

创建多区域接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
3. 选择要查看其配置详细信息的多区域接入点的名称。

   - 属性选项卡列出了与您的多区域接入点关联的所有桶、创建日期、Amazon 资源名称（ARN）和别名。AWS 账户 ID 列还列出了与多区域接入点关联的外部账户拥有的所有桶。
   - 权限选项卡列出了应用于与该多区域接入点相关联的桶的屏蔽公共访问权限设置。您还可以查看多区域接入点的多区域接入点策略（如果已创建）。权限页面上的信息提醒还列出了此多区域接入点的已启用屏蔽公共访问权限设置的所有桶（无论该桶是在您的账户中还是在其他账户中）。
   - 复制和失效转移选项卡提供与您的多区域接入点关联的桶以及桶所在区域的地图视图。如果有来自另一个账户的桶，但您无权从中提取数据，则该区域将在复制摘要地图上标记为红色，表明它是一个在获取复制状态时出错的 AWS 区域。

     > **Note**
     >
     > 要从外部账户中的桶检索复制状态信息，桶拥有者必须在其桶策略中向您授予 `s3:GetBucketReplication` 权限。

     此选项卡还提供与多区域接入点结合使用的区域的复制指标、复制规则和失效转移状态。

使用 AWS CLI

您可以使用 AWS CLI 查看多区域接入点的配置详细信息。

以下 AWS CLI 示例获取当前的多区域接入点配置。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-multi-region-access-point --account-id 111122223333 --name amzn-s3-demo-bucket1
```

删除多区域接入点

以下步骤介绍如何使用 Amazon S3 控制台删除多区域接入点。

删除多区域接入点并不会删除与多区域接入点相关联的桶，只会删除多区域接入点本身。

使用 S3 控制台

删除多区域接入点

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
3. 选择多区域接入点名称旁边的选项按钮。
4. 选择删除。
5. 在删除多区域接入点对话框中，输入要删除的 AWS 桶的名称。

> **Note**
>
> 确保输入有效的桶名称。否则，删除按钮将被禁用。

6. 选择删除以确认删除多区域接入点。

使用 AWS CLI

您可以使用 AWS CLI 删除多区域接入点。此操作不会删除与多区域接入点相关联的桶，只会删除多区域接入点本身。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control delete-multi-region-access-point --account-id 123456789012 --details
 Name=example-multi-region-access-point-name
```

配置多区域接入点以便与 AWS PrivateLink 共同使用

您可以使用多区域接入点在 AWS 区域之间路由 Amazon S3 请求流量。每个多区域接入点全局端点可路由来自多个来源的 Amazon S3 数据请求流量，而无需使用单独的端点构建复杂的网络配置。这些数据请求流量来源包括：

- 源自虚拟私有云（VPC）的流量
- 通过 AWS PrivateLink 传输的来自本地数据中心的流量
- 来自公共互联网的流量

如果您与 S3 多区域接入点建立 AWS PrivateLink 连接，则可以使用简单的网络架构和配置，通过私有连接将 S3 请求路由到 AWS 或跨多个 AWS 区域路由请求。使用 AWS PrivateLink 时，无需配置 VPC 对等连接。

主题

配置多区域接入点以便与 AWS PrivateLink 共同使用

使用虚拟私有云（VPC）中的私有 IP 地址，AWS PrivateLink 为您提供与 Amazon S3 的私有连接。您可以在 VPC 内预置一个或多个接口端点，连接到 Amazon S3 多区域接入点。

通过 AWS Management Console、AWS CLI 或 AWS 开发工具包，您可以创建多区域接入点的 com.amazonaws.s3-global.accesspoint 端点。要了解有关如何为多区域接入点配置接口端点的更多信息，请参阅 VPC 用户指南中的接口 VPC 端点。

要通过接口端点向多区域接入点发出请求，请按照以下步骤配置 VPC 和多区域接入点。

配置多区域接入点与 AWS PrivateLink 一起使用

1. 创建或拥有可连接到多区域接入点的适当 VPC 端点。有关创建 VPC 端点的更多信息，请参阅 VPC 用户指南中的接口 VPC 端点。

> **Important**
>
> 请确保创建 com.amazonaws.s3-global.accesspoint 端点。其他端点类型无法访问多区域接入点。

创建此 VPC 端点后，如果为端点启用了私有 DNS，则 VPC 中的所有多区域接入点请求都会通过此端点路由。默认为启用状态。

2. 如果多区域接入点策略不支持来自 VPC 端点的连接，需要对其进行更新。
3. 验证各个桶策略是否允许访问多区域接入点的用户。

请记住，多区域接入点的工作方式是将请求路由到桶，而不是自行完成请求。重要的是要牢记，因为请求的发起人必须具有对多区域接入点的权限，并允许访问多区域接入点中的各个桶。否则，请求可能会被路由到发起方没有执行请求的权限的桶。多区域接入点和关联的桶必须由相同或另一个 AWS 账户拥有。但是，如果权限配置正确，来自不同帐户的 VPC 可以使用多区域接入点。

因此，VPC 端点策略必须允许访问多区域接入点和您希望能够满足请求的每个底层桶。例如，假设您有一个别名为 `mfzwi23gnjvgw.mrap` 的多区域接入点。它由桶 `amzn-s3-demo-bucket1` 和 `amzn-s3-demo-bucket2` 提供支持，所有这些都归 AWS 账户 `123456789012` 所有。在这种情况下，以下 VPC 端点策略将允许来自 VPC 的对 `mfzwi23gnjvgw.mrap` 发出的 `GetObject` 请求由任一备份桶来完成。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Read-buckets-and-MRAP-VPCE-policy",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "arn:aws:s3:::amzn-s3-demo-bucket2/*",
            "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
        ]
    }]
```

```
}
```

如前所述，您还必须确保将多区域接入点策略配置为支持通过 VPC 端点进行访问。您无需指定正在请求访问的 VPC 端点。以下示例策略将向任何试图使用多区域接入点进行 GetObject 请求的请求者授予访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Sid": "Open-read-MRAP-policy",
        "Effect": "Allow",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
          ],
        "Resource": "arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap/object/*"
    }]
}
```

当然，每个桶都需要一个策略来支持通过 VPC 端点提交请求的访问。以下示例策略向任何匿名用户授予读取权限，其中包括通过 VPC 端点发出的请求。

```
{
    "Version":"2012-10-17",
   "Statement": [
   {
       "Sid": "Public-read",
       "Effect":"Allow",
       "Principal": "*",
       "Action": "s3:GetObject",
       "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket1",
           "arn:aws:s3:::amzn-s3-demo-bucket2/*"]
    }]
}
```

有关编辑 VPC 端点策略的更多信息，请参阅《VPC 用户指南》中的使用 VPC 端点控制对服务的访问。

从 VPC 端点删除对多区域接入点的访问

如果您拥有多区域接入点，并希望从接口端点删除对该接入点的访问权限，则必须为多区域接入点提供新的访问策略，以防止访问通过 VPC 端点的请求。但是，如果您的多区域接入点中的桶支持通过 VPC 端点的请求，它们将继续支持这些请求。如果想阻止该支持，还必须更新桶的策略。向多区域接入点提供新的访问策略只会阻止访问多区域接入点，而不会阻止访问底层桶。

> **Note**
>
> 您无法删除多区域接入点的访问策略。要删除对多区域接入点的访问权限，必须提供一个新的访问策略，其中包含所需的修改访问权限。

您可以更新桶策略来阻止通过 VPC 端点的请求，而不是更新多区域接入点的访问策略。在这种情况下，用户仍然可以通过 VPC 端点访问多区域接入点。但是，如果多区域接入点请求路由到某个桶，而该桶中的桶策略阻止访问，则该请求将生成错误消息。

通过多区域接入点发出请求

像其他资源一样，Amazon S3 多区域接入点具有 Amazon 资源名称（ARN）。您可以使用这些 ARN，通过 AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 API 将请求定向到多区域接入点。您还可以使用这些 ARN 在访问控制策略中识别多区域接入点。多区域接入点 ARN 不包括或披露多区域接入点的名称。有关 ARN 的更多信息，请参阅《AWS 一般参考》中的 Amazon 资源名称（ARN）。

> **Note**
>
> 多区域接入点别名和 ARN 不能互换使用。

多区域接入点 ARN 使用以下格式：

`arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

以下是多区域接入点 ARN 的几个示例：

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap` 表示别名为 `mfzwi23gnjvgw.mrap` 的多区域接入点，它由 AWS 账户 `123456789012` 拥有。

- arn:aws:s3::123456789012:accesspoint/* 表示账户 123456789012 下的所有多区域接入点。此 ARN 匹配账户 123456789012 的所有多区域接入点，但不匹配任何区域 Amazon S3 接入点，因为 ARN 不包括 AWS 区域。相比之下，ARN arn:aws:s3:us-west-2:123456789012:accesspoint/* 匹配账户 123456789012 下区域 us-west-2 中的所有区域 Amazon S3 接入点，但不匹配任何多区域接入点。

通过多区域接入点访问的对象的 ARN 采用以下格式：

arn:aws:s3::*account_id*:accesspoint/*MultiRegionAccessPoint_alias*//*key*

与多区域接入点 ARN 一样，通过多区域接入点访问的对象的 ARN 不包括 AWS 区域。下面是一些示例。

- arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01 表示 -01，可通过别名为 mfzwi23gnjvgw.mrap 的多区域接入点访问，由账户 123456789012 拥有。
- arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//* 表示账户 123456789012 中可以通过别名为 mfzwi23gnjvgw.mrap 的多区域接入点访问的所有对象。
- arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01/finance/* 表示账户 123456789012 中可以在别名为 mfzwi23gnjvgw.mrap 的多区域接入点的 -01/finance/ 下访问的所有对象。

多区域接入点主机名

您可以通过多区域接入点使用多区域接入点的主机名访问 Amazon S3 中的数据。可以将请求从公共互联网定向到该主机名。如果已为多区域接入点配置了一个或多个互联网网关，则请求也可以从虚拟私有云（VPC）定向到此主机名。有关创建与多区域接入点使用 VPC 接口端点的更多信息，请参阅 配置多区域接入点以便与 AWS PrivateLink 共同使用。

要使用 VPC 端点通过多区域接入点从 VPC 发出请求，可以使用 AWS PrivateLink。当您使用 AWS PrivateLink 对多区域接入点发出请求时，不能直接使用以 *region*.vpce.amazonaws.com 结尾的端点特定的区域域名系统（DNS）。此主机名将没有与其关联的证书，因此无法直接使用。您仍可以将 VPC 端点的公有域名系统（DNS）名称用作 CNAME 或 ALIAS 目标。或者，您可以在端点上启用私有域名系统（DNS），并使用标准的多区域接入点 *MultiRegionAccessPoint_alias*.accesspoint.s3-global.amazonaws.com 域名系统（DNS）名称，如本节所述。

通过多区域接入点向 API 请求 Amazon S3 数据操作时（例如 GetObject），请求的主机名如下所示：

MultiRegionAccessPoint_alias.accesspoint.s3-global.amazonaws.com

例如，要通过别名为 mfzwi23gnjvgw.mrap 的多区域接入点发出 GetObject 请求，请向主机名 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com 发出请求。此主机名的 s3-global 部分表明此主机名不适用于特定区域。

通过多区域接入点发出请求类似于通过单区域接入点发出请求。但是，务必注意以下差异：

- 多区域接入点 ARN 不包括 AWS 区域。他们遵循的格式 arn:aws:s3::*account-id*:accesspoint/*MultiRegionAccessPoint_alias*。
- 对于通过 API 操作发出的请求（这些请求不要求使用 ARN），多区域接入点使用不同的端点方案。模式是 *MultiRegionAccessPoint_alias*.accesspoint.s3-global.amazonaws.com — 例如，mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com。请注意与单区域接入点相比的差异：
  - 多区域接入点主机名使用其别名，而不是多区域接入点名称。
  - 多区域接入点主机名不包括拥有者的 AWS 账户 ID。
  - 多区域接入点主机名不包括 AWS 区域。
  - 多区域接入点主机名包括 s3-global.amazonaws.com 而不是 s3.amazonaws.com。
- 必须使用签名版本 4A（SigV4A）签署多区域接入点请求。使用 AWS SDK 时，SDK 会自动将 SigV4 转换为 SigV4A。因此，请确保您的 AWS SDK 支持将 SigV4A 作为用于签署全局 AWS 区域请求的签名实现。有关 SigV4A 的更多信息，请参阅《AWS 一般参考》中的签署 AWS API 请求。

多区域接入点和 Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration 是一项启用数据快速转换到桶的功能。Transfer Acceleration 是在单个桶级别上配置的。有关 Transfer Acceleration 的更多信息，请参阅使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输。

多区域接入点使用类似于 Transfer Acceleration 的加速传输机制，通过 AWS 网络发送大对象。因此，在通过多区域接入点发送请求时，您无需使用 Transfer Acceleration。这种提高的传输性能会自动融入到多区域接入点中。

主题

- 权限
- 多区域接入点限制和限制
- 多区域接入点请求路由

- Amazon S3 多区域接入点失效转移控制
- 配置用于多区域接入点的复制
- 将多区域接入点与支持的 API 操作结合使用
- 监视和记录通过多区域接入点发出的对底层资源的请求

权限

Amazon S3 多区域接入点可以简化多个 AWS 区域中 Amazon S3 桶的数据访问。多区域接入点称为全局端点，您可以使用这些接入点执行 Amazon S3 数据接入点操作，如 `GetObject` 和 `PutObject`。对于通过全局端点发出的任何请求，每个多区域接入点都可以具有不同的权限和网络控制。

每个多区域接入点也可以强制实施自定义访问策略，该策略与附加到底层桶的桶策略结合使用。要使请求成功，以下所有条件都必须允许该操作：

- 多区域接入点策略
- 底层 AWS Identity and Access Management（IAM）策略
- 底层桶策略（请求路由到哪里）

您可以将任何多区域接入点策略配置为仅接受来自特定 IAM 用户或组的请求。有关如何执行此操作的示例，请参阅the section called “多区域接入点策略示例”中的示例 2。为了限制 Amazon S3 数据访问私有网络，您可以将任何多区域接入点策略配置为仅接受来自虚拟私有云（VPC）的请求。

例如，假设您通过多区域接入点使用您的 AWS 账户中名为 `AppDataReader` 的用户发出一个 `GetObject` 请求。为了帮助确保请求不会被拒绝，`AppDataReader` 用户必须被多区域接入点和多区域接入点底层的每个桶授予 `s3:GetObject` 权限。`AppDataReader` 将无法从任何未授予此权限的任何桶中检索数据。

> **Important**
>
> 通过桶名称或 Amazon 资源名称（ARN）直接访问桶时，将桶的访问控制委派给多区域接入点策略不会改变桶的行为。直接对桶进行的所有操作将继续像以前一样运行。您在多区域接入点策略中包括的限制仅适用于通过该多区域接入点发出的请求。

管理多区域接入点的公共访问

多区域接入点支持每个多区域接入点的独立阻止公有访问设置。创建多区域接入点时，您可以指定应用于该多区域接入点的阻止公有访问设置。

> **Note**
>
> 即使多区域接入点的独立屏蔽公共访问权限设置被禁用，在此账户的屏蔽公共访问权限设置（在您自己的账户中）或外部桶的屏蔽公共访问权限设置下启用的任何屏蔽公共访问权限设置也仍然适用。

对于通过多区域接入点发出的任何请求，Amazon S3 会评估以下各项的阻止公有访问设置：

- 多区域接入点
- 底层桶（包括外部桶）
- 拥有多区域接入点的账户
- 拥有底层桶的账户（包括外部账户）

如果这些设置中的任何一个指示应阻止请求，则 Amazon S3 会拒绝请求。有关 Amazon S3 阻止公有访问功能的更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

> **Important**
>
> 原定设置情况下，为多区域接入点启用所有阻止公有访问设置。您必须明确禁用不希望应用于多区域接入点的任何设置。
> 在创建多区域接入点之后，您无法更改其屏蔽公共访问权限设置。

查看多区域接入点的阻止公有访问设置

查看多区域接入点的阻止公有访问设置

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。

3. 请选择要查看的多区域接入点的名称。
4. 选择权限选项卡。
5. 在 Block Public Access settings for this Multi-Region Access Point（多区域接入点的阻止公有访问设置）下，查看多区域接入点的阻止公有访问设置。

> **Note**
>
> 创建多区域接入点后，无法编辑阻止公有访问设置。因此，如果您要阻止公有访问，请在创建多区域接入点之前，确保您的应用程序在没有公有访问的情况下正常运行。

使用多区域接入点策略

以下示例多区域接入点策略授予 IAM 用户从多区域接入点列出和下载文件的访问权限。要使用此示例策略，请将 *user input placeholders* 替换为您自己的信息。

```
{
  "Version":"2012-10-17",
  "Statement":[
     {
        "Effect":"Allow",
        "Principal":{
           "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
        },
        "Action":[
           "s3:ListBucket",
           "s3:GetObject"
        ],
        "Resource":[
           "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
           "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/
*"
        ]
     }
  ]
}
```

要使用 AWS Command Line Interface（AWS CLI）将多区域接入点策略与指定的多区域接入点关联，请使用以下 `put-multi-region-access-point-policy` 命令。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。每个多区域接入点只能有一个策略，因此，对 `put-`

multi-region-access-point-policy 操作发出的请求将替换与指定多区域接入点关联的任何现有策略。

AWS CLI

```
aws s3control put-multi-region-access-point-policy
--account-id 111122223333
--details { "Name": "amzn-s3-demo-bucket-MultiRegionAccessPoint",
 "Policy": "{ \"Version\": \"2012-10-17\", \"Statement\": { \"Effect\":
 \"Allow\", \"Principal\": { \"AWS\": \"arn:aws:iam::111122223333:root
\" }, \"Action\": [\"s3:ListBucket\", \"s3:GetObject\"], \"Resource\":
 [ \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
 \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/*
\" ] } }" }
```

要查询上次操作的结果，请使用以下命令：

AWS CLI

```
aws s3control describe-multi-region-access-point-operation
--account-id 111122223333
--request-token-arn requestArn
```

要检索多区域接入点策略，请使用以下命令：

AWS CLI

```
aws s3control get-multi-region-access-point-policy
--account-id 111122223333
--name=amzn-s3-demo-bucket-MultiRegionAccessPoint
```

编辑多区域接入点策略

多区域接入点策略（以 JSON 编写）提供对与多区域接入点结合使用的 Amazon S3 桶的存储访问。您可以允许或拒绝特定主体对您的多区域接入点执行各种操作。当一个请求通过多区域接入点路由到一个桶时，多区域接入点和桶的访问策略都适用。限制性更强的访问策略始终优先。

> **Note**
>
> 如果桶包含由其他账户拥有的对象，则多区域接入点策略不适用于由其他 AWS 账户拥有的对象。

应用多区域接入点策略后，无法删除该策略。您可以编辑策略，也可以创建覆盖现有策略的新策略。

编辑多区域接入点策略

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
3. 选择要为其编辑策略的多区域接入点的名称。
4. 选择权限选项卡。
5. 向下滚动到 Multi-Region Access Point policy（多区域接入点策略）部分。选择 Edit（编辑）以更新策略（采用 JSON 格式）。
6. 将显示 Edit Multi-Region Access Point policy（编辑多区域接入点策略）页面。您可以直接在文本字段中输入策略，也可以选择 Add statement（添加语句）以从下拉列表中选择策略元素。

> **Note**
>
> 控制台会自动显示您可以在策略中使用的多区域接入点 Amazon 资源名称（ARN）。有关多区域接入点策略示例，请参阅 the section called “多区域接入点策略示例”。

多区域接入点策略示例

Amazon S3 多区域接入点支持 AWS Identity and Access Management（IAM）资源策略。您可以使用这些策略，按资源、用户或其他条件控制多区域接入点的使用。为了使应用程序或用户能够通过多区域接入点访问对象，多区域接入点和底层桶都必须允许相同的访问权限。

要允许对多区域接入点和底层桶进行相同访问，请执行以下操作之一：

- （推荐）要简化使用 Amazon S3 多区域接入点时的访问控制，请将 Amazon S3 桶的访问控制权委托给多区域接入点。有关如何执行此操作的示例，请参阅本节中的示例 1。
- 将多区域接入点策略中包含的相同权限添加到底层桶策略中。

> **Important**
>
> 通过桶名称或 Amazon 资源名称（ARN）直接访问桶时，将桶的访问控制委派给多区域接入点策略不会改变桶的行为。直接对桶进行的所有操作将继续像以前一样运行。您在多区域接入点策略中包括的限制仅适用于通过该多区域接入点发出的请求。

Example 1 – 将访问权限委托给桶策略中的特定多区域接入点（针对同一账户或跨账户）

以下示例桶策略授予对特定多区域接入点的完全桶访问权限。这意味着，对此桶的所有访问都由附加到多区域接入点的策略控制。我们建议您以这种方式为所有不需要直接访问存储桶的使用案例配置存储桶。您可以将此桶策略结构用于同一账户或其他账户中的多区域接入点。

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN", "Bucket ARN/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointArn" : "MultiRegionAccessPoint_ARN" }
        }
    }]
}
```

> **Note**
>
> 如果您正在授予对多个多区域接入点的访问权限，请确保列出每个多区域接入点。

Example 2 – 在您的多区域接入点策略中授予账户对多区域接入点的访问权限

以下多区域接入点策略向账户 *123456789012* 授予列出和读取由 *MultiRegionAccessPoint_ARN* 定义的多区域接入点中包含的对象的权限。

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
       "Effect": "Allow",
       "Principal": {
          "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
       },
       "Action":[
          "s3:ListBucket",
          "s3:GetObject"
       ],
       "Resource":[
          "MultiRegionAccessPoint_ARN",
          "MultiRegionAccessPoint_ARN/object/*"
       ]
     }
  ]
}
```

Example 3 – 允许列出桶内容的多区域接入点策略

以下多区域接入点策略向账户 *123456789012* 授予列出由 *MultiRegionAccessPoint_ARN* 定义的多区域接入点中包含的对象的权限。

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
       "Effect": "Allow",
       "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/JohnDoe"
        },
        "Action": "s3:ListBucket",
        "Resource": "MultiRegionAccessPoint_ARN"
       }
    ]
}
```

多区域接入点限制和限制

Amazon S3 多区域接入点具有以下限制和局限性：

- 多区域接入点名称：
  - 在单个 AWS 账户内必须是唯一的

  - 必须以数字或小写字母开头
  - 长度必须介于 3-50 个字符之间
  - 不能以连字符（-）开头或结尾
  - 不能包含下划线（_）、大写字母或句点（.）
  - 创建后无法对其进行编辑
- 多区域接入点别名由 Amazon S3 生成，无法编辑或重复使用。
- 您无法使用网关端点通过多区域接入点访问数据。但是，您可以使用接口端点通过多区域接入点访问数据。要使用 AWS PrivateLink，您必须创建 VPC 端点。有关更多信息，请参阅 配置多区域接入点以便与 AWS PrivateLink 共同使用。
- 要在 Amazon CloudFront 中使用多区域接入点，必须将多区域接入点配置为 `Custom Origin` 分配类型。有关各种源类型的更多信息，请参阅在 CloudFront 分配中使用各种源。有关在 Amazon CloudFront 中使用多区域接入点的更多信息，请参阅 AWS 存储博客上的跨多个区域构建主动-主动、基于距离远近的应用程序。
- 多区域接入点最低要求：
  - 传输层安全性 (TLS) v1.2
  - 签名版本 4 (SigV4A)

    多区域接入点支持签名版本 4A。此版本的 SigV4 允许对多个 AWS 区域签署请求。此特征对于可能导致从多个区域之一访问数据的 API 操作非常有用。使用 AWS SDK，您可以提供凭证，并且对多区域接入点的请求将使用签名版本 4A，而无需进行其他配置。请务必检查您的 AWS SDK 与 SigV4A 算法的兼容性。有关 SigV4A 的更多信息，请参阅《AWS 一般参考》中的签署 AWS API 请求。

    > **Note**
    >
    > 要将 SigV4A 与临时安全凭证一起使用 [例如，在使用 AWS Identity and Access Management（IAM）角色时]，您可以从区域 AWS Security Token Service（AWS STS）端点请求临时凭证。如果您从全局 AWS STS 端点（`sts.amazonaws.com`）请求临时凭证，则必须先将全局端点会话令牌的区域兼容性设置为在所有 AWS 区域中都有效。有关更多信息，请参阅《IAM 用户指南》中的在 AWS 区域中管理 AWS STS。
- 多区域接入点不支持匿名请求。
- 多区域接入点限制：
  - 不支持 IPv6。

  - 不支持 Outposts 桶上面的 Amazon S3。
  - 当使用多区域接入点 ARN 时，多区域接入点仅支持使用多区域接入点的复制操作作为目标。
  - 不支持 S3 批量操作特征。
- 不支持某些 AWS SDK。要确认多区域接入点支持哪些 AWS SDK，请参阅与 AWS SDK 的兼容性。
- 多区域接入点的服务限额如下所示：
  - 每个账户最多有 100 个多区域接入点。
  - 单个多区域接入点的限制为 17 个区域。
- 创建多区域接入点后，您无法从多区域接入点配置中添加、修改或删除桶。要更改桶，您必须删除整个多区域接入点，然后创建新的接入点。如果删除了多区域接入点中的跨账户桶，则重新连接此桶的唯一方法是使用该账户中的相同名称和区域重新创建该桶。
- 只有在删除多区域接入点后，才能删除多区域接入点中使用的底层桶（在同一账户中）。
- 所有创建或维护多区域接入点的控制面板请求都必须路由到 US West (Oregon) 区域。对于多区域接入点数据面板请求，无需指定区域。
- 对于多区域接入点失效转移控制面板，必须将请求路由到以下五个支持的区域之一：
  - US East (N. Virginia)
  - US West (Oregon)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Europe (Ireland)
- 多区域接入点仅支持以下 AWS 区域中的桶：
  - US East (N. Virginia)
  - US East (Ohio)
  - US West (N. California)
  - US West (Oregon)
  - Asia Pacific (Mumbai)
  - Asia Pacific (Osaka)
  - Asia Pacific (Seoul)
  - Asia Pacific (Singapore)
  - Asia Pacific (Sydney)
  - Asia Pacific (Tokyo)
  - Canada (Central)

- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- South America (São Paulo)

多区域接入点请求路由

通过多区域接入点发出请求时，Amazon S3 会确定与多区域接入点相关联的桶中的哪一个桶距离您最近。然后，Amazon S3 将请求定向到该桶，而不考虑 AWS 它所在的区域。

在多区域接入点将请求路由到距离最近的桶之后，Amazon S3 会像您直接向该桶发出请求一样处理请求。多区域接入点不了解 Amazon S3 桶的数据内容。因此，收到请求的桶可能不包含所请求的数据。要在 Amazon S3 桶中创建与多区域接入点相关联的一致数据集，您可以配置 S3 跨区域复制（CRR）。然后任何桶都可以成功满足 请求。

Amazon S3 根据以下规则导向多区域接入点请求：

- Amazon S3 根据距离远近优化要完成的请求。它查看多区域接入点支持的桶，并将请求中继到距离最近的桶。
- 如果请求指定了现有资源（例如，`GetObject`），Amazon S3 在满足请求时不会考虑对象的名称。这意味着，即使多区域接入点中的一个桶中存在某个对象，您的请求也可能路由到不包含该对象的桶。这种情况会导致向客户端返回 404 错误消息。

  为了避免 404 错误，我们建议您为桶配置 S3 跨区域复制（CRR）。当您需要的对象位于多区域接入点中的桶中，但不在您的请求路由到的特定桶中时，复制有助于解决潜在问题。有关复制配置的信息，请参阅 配置用于多区域接入点的复制。

  为了确保使用所需的特定对象满足您的请求，我们还建议您启用桶版本控制并在请求中包含版本 ID。这种方法有助于确保您具有您正在查找的对象的正确版本。启用了版本控制的桶还可以帮助您恢复意外覆盖的对象。有关更多信息，请参阅在 S3 桶中使用 S3 版本控制。
- 如果请求是创建一个资源（例如，`PutObject` 或 `CreateMultipartUpload`），则 Amazon S3 将使用距离最近的桶完成请求。例如，假设一家视频公司希望支持从世界任何地方上传视频。当用户将 PUT 请求发送到多区域接入点时，该对象将放入距离最近的桶中。然后，要以最低的延迟将上传的视频提供给世界各地的其他人下载，您可以将 CRR 与双向复制结合使用。将 CRR 与双向复制结

合使用，可使与多区域接入点关联的所有桶的内容保持同步。有关将复制与多区域接入点结合使用的更多信息，请参阅配置用于多区域接入点的复制。

Amazon S3 多区域接入点失效转移控制

借助 Amazon S3 多区域接入点失效转移控制，您可以在区域流量中断期间保持业务连续性，同时还可以为您的应用程序提供多区域架构以满足合规性和冗余需求。如果您的区域流量中断，您可以使用多区域接入点失效转移控制来选择 Amazon S3 多区域接入点后面的哪个 AWS 区域将处理数据访问和存储请求。

要支持失效转移，您可以将多区域接入点设置为主动-被动配置，在正常情况下流量流向主动区域，而被动区域处于备用状态以用于失效转移。

例如，要执行到您选择的某个 AWS 区域的失效转移，您需要将流量从主要（主动）区域转移到辅助（被动）区域。在这样的主动-被动配置中，一个桶处于主动状态并接受流量，而另一个桶是被动的，不接受流量。被动桶用于灾难恢复。当您启动失效转移时，所有流量（例如 GET 或 PUT 请求）都将定向到处于主动状态的桶（位于一个区域中），并远离处于被动状态的桶（位于另一个区域中）。

如果您通过双向复制规则启用 S3 跨区域复制（CRR），则可以在失效转移期间保持桶同步。此外，如果您在主动-主动配置中启用了 CRR，Amazon S3 多区域接入点还可以从距离最近的桶位置提取数据，从而提高应用程序性能。

AWS 区域 支持

借助 Amazon S3 多区域接入点失效转移控制，您的 S3 桶可以位于支持多区域接入点的 17 个区域中的任何一个区域。您可以一次跨任意两个区域启动失效转移。

> **Note**
>
> 尽管一次只能在两个区域之间启动失效转移，但您可以在多区域接入点中同时单独更新多个区域的路由状态。

以下主题演示如何使用和管理 Amazon S3 多区域接入点失效转移控制。

主题

- Amazon S3 多区域接入点路由状态
- 使用 Amazon S3 多区域接入点失效转移控制
- Amazon S3 多区域接入点失效转移控制错误

Amazon S3 多区域接入点路由状态

您的 Amazon S3 多区域接入点失效转移配置决定了与多区域接入点结合使用的 AWS 区域的路由状态。您可以将 Amazon S3 多区域接入点配置为主动-主动状态或主动-被动状态。

- 主动-主动 – 在主动-主动配置中，所有请求都会自动发送到多区域接入点中距离最近的 AWS 区域。将多区域接入点配置为活动-活动状态后，所有区域都可以接收流量。如果在主动-主动配置中发生流量中断，网络流量将自动重定向到其中一个活动的区域。
- 主动-被动 – 在主动-被动配置中，多区域接入点中的主动区域接收流量，而被动区域不接收流量。如果您打算在灾难情况下使用 S3 失效转移控制启动失效转移，请在测试和执行灾难恢复计划时以主动-被动配置设置您的多区域接入点。

使用 Amazon S3 多区域接入点失效转移控制

本部分介绍如何通过 AWS Management Console来管理和使用 Amazon S3 多区域接入点失效转移控制。

在 AWS Management Console中的多区域接入点详细信息页面上的 Failover configuration（失效转移配置）部分中有两个失效转移控制：Edit routing status（编辑路由状态）和 Failover（失效转移）。您可以使用这些控件执行以下操作：

- Edit routing status（编辑路由状态）- 通过选择 Edit routing status（编辑路由状态），您可以在多区域接入点的单个请求中手动编辑最多 17 个 AWS 区域的路由状态。您可以将 Edit routing status（编辑路由状态）用于以下目的：
  - 设置或编辑多区域接入点中一个或多个区域的路由状态
  - 通过将两个区域配置为主动-被动状态来为您的多区域接入点创建失效转移配置
  - 手动失效转移您的区域
  - 在区域之间手动切换流量
- Failover（失效转移）– 当您选择 Failover（失效转移）以启动失效转移时，您只更新已配置为主动-被动状态的两个区域的路由状态。在通过选择 Failover（失效转移）启动的失效转移期间，两个区域之间的路由状态将自动切换。

编辑多区域接入点中区域的路由状态

通过在多区域接入点详细信息页面的 Failover configuration（失效转移配置）部分中选择 Edit routing status（编辑路由状态），您可以在多区域接入点的单个请求中手动更新最多 17 个 AWS 区域的路由状态。但是，当您通过选择 Failover（失效转移）启动失效转移时，您只更新已配置为主动-被动状态

的两个区域的路由状态。在通过选择 Failover（失效转移）启动的失效转移期间，两个区域之间的路由状态将自动切换。

您可以将 Edit routing status（编辑路由状态）（如以下步骤所述）用于以下目的：

- 设置或编辑多区域接入点中一个或多个区域的路由状态
- 通过将两个区域配置为主动-被动状态来为您的多区域接入点创建失效转移配置
- 手动失效转移您的区域
- 在区域之间手动切换流量

使用 S3 控制台

更新多区域接入点中区域的路由状态

1. 登录 AWS 管理控制台。
2. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
3. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
4. 请选择要更新的多区域接入点。
5. 选择 Replication and failover（复制和失效转移）选项卡。
6. 请选择要编辑其路由状态的一个或多个区域。

> **Note**
>
> 要启动失效转移，必须在多区域接入点中将至少一个 AWS 区域指定为 Active（主动），并将一个区域指定为 Passive（被动）。

7. 选择 Edit routing status（编辑路由状态）。
8. 在出现的对话框中，为每个区域的 Routing status（路由状态）选择 Active（主动）或 Passive（被动）。

   活动状态允许将流量路由到该区域。被动状态会阻止任何流量定向到该区域。

   如果您正在为多区域接入点创建失效转移配置或启动失效转移，必须在多区域接入点中将至少一个 AWS 区域指定为 Active（主动），并将一个区域指定为 Passive（被动）。
9. 选择 Save routing status（保存路由状态）。重定向流量大约需要 2 分钟。

在对于多区域接入点提交 AWS 区域的路由状态后，您可以验证路由状态的变化。要验证这些变化，请访问 Amazon CloudWatch（网址为 https://console.aws.amazon.com/cloudwatch/）以监控您的 Amazon S3 数据请求流量（例如 GET 和 PUT 请求）在主动和被动区域之间的转移。在失效转移期间不会终止任何现有连接。现有连接将一直持续到它们达到成功或失败状态。

使用 AWS CLI

> **Note**
>
> 您可以对以下五个区域中的任何一个运行多区域接入点 AWS CLI 路由命令：
>
> - ap-southeast-2
> - ap-northeast-1
> - us-east-1
> - us-west-2
> - eu-west-1

以下示例命令更新您当前的多区域接入点路由配置。要更新桶的主动或被动状态，请对于主动将 TrafficDialPercentage 值设置为 100，对于被动则设置为 0。在此示例中，*DOC-EXAMPLE-BUCKET-1* 设置为主动，*DOC-EXAMPLE-BUCKET-2* 设置为被动。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
--route-updates Bucket=DOC-EXAMPLE-BUCKET-1,TrafficDialPercentage=100
                Bucket=DOC-EXAMPLE-BUCKET-2,TrafficDialPercentage=0
```

以下示例命令获取更新的多区域接入点路由配置。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
```

启动失效转移

当您通过在多区域接入点详细信息页面上的 Failover configuration（失效转移配置）部分中选择 Failover（失效转移）来启动失效转移时，Amazon S3 请求流量会自动转移到备用 AWS 区域。失效转移过程在 2 分钟内完成。

您可以一次跨任意两个 AWS 区域（属于支持多区域接入点的 17 个区域）启动失效转移。然后，在 AWS CloudTrail 中记录失效转移事件。失效转移完成后，您可以在 Amazon CloudWatch 中监控 Amazon S3 流量和对于新活动区域的任何流量路由更新。

> **Important**
>
> 要在数据复制期间使所有元数据和对象在桶之间保持同步，我们建议您在配置失效转移控制之前创建双向复制规则并启用副本修改同步。
> 双向复制规则有助于确保在将数据写入流量失效转移到的 Amazon S3 桶时，该数据随后被复制回源桶。副本修改同步有助于确保在双向复制期间，对象元数据也在桶之间同步。
> 有关配置复制以支持失效转移的更多信息，请参阅the section called “桶复制”。

在复制的桶之间启动失效转移

1. 登录 AWS 管理控制台。
2. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
3. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
4. 请选择要用于启动失效转移的多区域接入点。
5. 选择 Replication and failover（复制和失效转移）选项卡。
6. 向下滚动到 Failover configuration（失效转移配置）部分，然后选择两个 AWS 区域。

   > **Note**
   >
   > 要启动失效转移，必须在多区域接入点中将至少一个 AWS 区域指定为 Active（主动），并将一个区域指定为 Passive（被动）。活动状态允许将流量定向到区域。被动状态会阻止任何流量定向到该区域。

7. 选择 Failover（故障转移）。
8. 在此对话框中，再次选择 Failover（失效转移）以启动失效转移过程。在此过程中，两个区域的路由状态会自动切换。所有新流量都定向到变为活动状态的区域，而流量将停止定向到变为被动状态的区域。重定向流量大约需要 2 分钟。

启动失效转移过程后，您可以验证流量变化。要验证这些变化，请访问 Amazon CloudWatch（网址为 https://console.aws.amazon.com/cloudwatch/）以监控您的 Amazon S3 数据请求流量（例如 GET 和 PUT 请求）在主动和被动区域之间的转移。在失效转移期间不会终止任何现有连接。现有连接将一直持续到它们达到成功或失败状态。

查看您的 Amazon S3 多区域接入点路由控制

使用 S3 控制台

查看您的 Amazon S3 多区域接入点的路由控制

1. 登录 AWS 管理控制台。
2. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
3. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
4. 请选择要查看的多区域接入点。
5. 选择 Replication and failover（复制和失效转移）选项卡。此页面显示您的多区域接入点的路由配置详细信息和摘要、关联的复制规则和复制指标。您可以在 Failover configuration（失效转移配置）部分查看区域的路由状态。

使用 AWS CLI

以下示例 AWS CLI 命令获取指定区域的当前多区域接入点路由配置。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap MultiRegionAccessPoint_ARN
```

> **Note**
>
> 此命令只能对以下五个区域执行：
>
> - ap-southeast-2
> - ap-northeast-1
> - us-east-1

- us-west-2
- eu-west-1

Amazon S3 多区域接入点失效转移控制错误

更新多区域接入点的失效转移配置时，可能会遇到以下错误之一：

- HTTP 400 错误请求：如果您在更新失效转移配置时输入了无效的多区域接入点 ARN，就会发生此错误。您可以通过查看多区域接入点策略来确认多区域接入点 ARN。要查看或更新多区域接入点策略，请参阅编辑多区域接入点策略。如果您在更新 Amazon S3 多区域接入点失效转移控制时使用空字符串或随机字符串，也会发生此错误。确保使用多区域接入点 ARN 格式：

  `arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

- HTTP 503 Slow Down（HTTP 503 减慢速度）：如果您在短时间内发送的请求过多，就会出现此错误。被拒绝的请求将导致错误。
- HTTP 409 Conflict（HTTP 409 冲突）：当两个或多个并发的路由配置更新请求以单个多区域接入点为目标时，就会出现此错误。第一个请求成功，但任何其他请求都失败而出现错误。
- HTTP 405 Method Not Allowed（HTTP 405 不允许方法）：当您在启动失效转移时选择了只有一个 AWS 区域的多区域接入点时，就会出现此错误。在启动失效转移之前，必须选择两个区域。否则，将返回错误。

配置用于多区域接入点的复制

向多区域接入点端点发出请求时，Amazon S3 会自动将请求路由到离您最近的桶。Amazon S3 在做出这一决定时不考虑请求的内容。如果发出请求以 GET 对象，您的请求可能会被路由到没有此对象副本的桶。如果发生这种情况，您将收到 HTTP 状态代码 404（找不到）错误。有关多区域接入点请求路由的更多信息，请参阅the section called “请求路由”。

如果您希望多区域接入点能够检索对象，而无论哪个桶接收请求，则必须配置 Amazon S3 跨区域复制（CRR）。

例如，考虑具有三个桶的多区域接入点：

- 区域 us-west-2 中包含对象 my-image.jpg 的名为 my-bucket-usw2 的桶
- 区域 ap-south-1 中包含对象 my-image.jpg 的名为 my-bucket-aps1 的桶

- 区域 eu-central-1 中不包含对象 my-image.jpg 且名为 my-bucket-euc1 的桶

在这种情况下，如果您向 GetObject 对象发出请求 my-image.jpg，则该请求的成功取决于接收您请求的桶。由于 Amazon S3 不考虑请求的内容，因此它可能会将 GetObject 请求添加到 my-bucket-euc1 桶（如果该桶响应距离最近的请求）。即使您的对象位于多区域接入点中的桶中，因为收到您请求的单个桶没有该对象，您仍会收到 HTTP 404 Not Found（HTTP 404 找不到）错误。

启用跨区域复制（CRR）有助于避免这种结果。使用适当的复制规则，将 my-image.jpg 对象复制到 my-bucket-euc1 桶。因此，如果 Amazon S3 将您的请求路由到该桶，您现在可以检索该对象。

对于分配给多区域接入点的桶，复制工作正常。Amazon S3 不会对位于多区域接入点中的桶执行任何特殊复制。有关在桶中配置复制的更多信息，请参阅 设置实时复制。

对多区域接入点使用复制的建议

为了在使用多区域接入点时获得最佳的复制性能，我们建议采取以下措施：

- 配置 S3 Replication Time Control（S3 RTC）。要在可预测的时间范围内跨不同区域复制您的数据，您可以使用 S3 RTC。S3 RTC 在 15 分钟内复制 Amazon S3 中存储的 99.99% 的新对象（由服务等级协议提供支持）。有关更多信息，请参阅the section called “使用 S3 Replication Time Control”。使用 S3 RTC 无需额外付费。有关信息，请参阅 Amazon S3 定价。
- 使用双向复制，以支持在通过多区域接入点更新桶时保持桶同步。有关更多信息，请参阅the section called “为多区域接入点创建双向复制规则”。
- 创建跨账户多区域接入点，以将数据分别复制到单独 AWS 账户中的桶。这种方法提供了账户级分离，这样，就可以从与源桶不同的区域中的不同账户访问数据以及跨不同账户复制数据。设置跨账户多区域接入点无需额外费用。如果您是桶拥有者但不拥有多区域接入点，则只需支付数据传输和请求费用。多区域接入点拥有者支付数据路由和互联网加速费用。有关更多信息，请参阅 Amazon S3 定价。
- 为每个复制规则启用副本修改同步，以使对于对象的元数据更改也保持同步。有关更多信息，请参阅启用副本修改同步。
- 启用 Amazon CloudWatch 指标来监控复制事件。将收取 CloudWatch 指标费用。有关更多信息，请参阅 Amazon CloudWatch 定价。

主题

- 为多区域接入点创建单向复制规则
- 为多区域接入点创建双向复制规则
- 查看多区域接入点的复制规则

为多区域接入点创建单向复制规则

复制规则允许跨桶自动和异步复制对象。单向复制规则有助于确保将数据从一个 AWS 区域中的源桶完全复制到另一个区域的目标桶。当设置单向复制时，将创建从源桶（DOC-EXAMPLE-BUCKET-1）到目标桶（DOC-EXAMPLE-BUCKET-2）的复制规则。与所有复制规则一样，您可以将单向复制规则应用于整个 Amazon S3 桶，也可以应用于按前缀或对象标签筛选的对象子集。

> **Important**
>
> 如果您的用户只使用目标桶中的对象，我们建议使用单向复制。如果您的用户要上传或修改目标桶中的对象，请使用双向复制来保持所有桶同步。如果您计划使用多区域接入点进行失效转移，我们也建议使用双向复制。要设置双向复制，请参阅the section called “为多区域接入点创建双向复制规则”。

为多区域接入点创建单向复制规则

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
3. 请选择多区域接入点的名称。
4. 选择 Replication and failover（复制和失效转移）选项卡。
5. 向下滚动到 Replication rules（复制规则）部分，然后选择 Create replication rules（创建复制规则）。确保您有足够的权限来创建复制规则，否则将禁用版本控制。

   > **Note**
   >
   > 您只能为自己账户中的桶创建复制规则。要为外部桶创建复制规则，桶拥有者必须为这些桶创建复制规则。

6. 在创建复制规则页面上，选择将对象从一个或多个源桶复制到一个或多个目标桶模板。

   > **Important**
   >
   > 当您使用此模板创建复制规则时，它们会替换已经分配给桶的任何现有复制规则。要添加或修改任何现有的复制规则而不是替换它们，请转到控制台中每个桶的 Management（管理）选项卡，然后在 Replication rules（复制规则）部分编辑规则。还

可以使用 AWS CLI、SDK 或 REST API 添加或修改现有复制规则。有关更多信息，请参阅复制配置。

7. 在源和目标部分中的源桶下，选择要从中复制对象的一个或多个桶。选择进行复制的所有桶（源和目标）都必须启用 S3 版本控制，并且每个桶必须位于不同的 AWS 区域中。有关 S3 版本控制的更多信息，请参阅在 Amazon S3 桶中使用版本控制。

   在目标桶下，选择要将对象复制到其中的一个或多个桶。

   > **Note**
   >
   > 确保您具有建立复制所需的读取和复制权限，否则会遇到错误。有关更多信息，请参阅创建 IAM 角色。

8. 在 Replication rule configuration（复制规则配置）部分中，选择在创建复制规则时将其设置为 Enabled（启用）还是 Disabled（禁用）。

   > **Note**
   >
   > 您无法在 Replication rule name（复制规则名称）框中输入名称。复制规则名称是根据您创建复制规则时的配置生成的。

9. 在 Scope（范围）部分中，为复制选择适当的范围。

   - 要复制整个桶，请选择 Apply to all objects in the bucket（应用到桶中的所有对象）。
   - 为了复制桶中对象的子集，请选择 Limit the scope of this rule using one or more filters（使用一个或多个筛选条件限制此规则的范围)。

     可按前缀、对象标签或二者的组合筛选对象。

     - 要将复制限制为名称以相同字符串（例如 `pictures`）开头的所有对象，请在 Prefix（前缀）框中输入前缀。

       如果输入的前缀是文件夹的名称，则必须使用分隔符 [如 /（正斜杠）] 来指示其层次级别（例如 `pictures/`）。有关前缀的更多信息，请参阅使用前缀整理对象。

     - 要复制具有一个或多个对象标签的所有对象，请选择 Add tag（添加标签），然后在框中输入键值对。要添加另一个标签，请重复该过程。有关对象标签的更多信息，请参阅 使用标签对存储进行分类。

10. 向下滚动到 Additional replication options（其他复制选项）部分，然后选择要应用的复制选项。

> **Note**
>
> 我们建议您应用以下选项：
>
> - Replication time control（RTC）[复制时间控制（RTC）] – 要在可预测的时间范围内跨不同区域复制您的数据，您可以使用 S3 Replication Time Control（S3 RTC）。S3 RTC 在 15 分钟内复制 Amazon S3 中存储的 99.99% 的新对象（由服务等级协议提供支持）。有关更多信息，请参阅the section called “使用 S3 Replication Time Control”。
> - Replication metrics and notifications（复制指标和通知）– 启用 Amazon CloudWatch 指标以监控复制事件。
> - 删除标记复制 - 将复制由 S3 删除操作创建的删除标记。不会复制由生命周期规则创建的删除标记。有关更多信息，请参阅在桶之间复制删除标记。
>
> S3 RTC 和 CloudWatch 复制指标和通知需要额外收费。有关更多信息，请参阅 Amazon S3 定价和 Amazon CloudWatch 定价。

11. 如果您正在编写取代现有复制规则的新复制规则，请选择 I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten（我确认通过选择“创建复制规则”，这些现有的复制规则将被覆盖）。
12. 选择创建复制规则以创建和保存新的单向复制规则。

为多区域接入点创建双向复制规则

复制规则允许跨桶自动和异步复制对象。双向复制规则可确保数据在不同 AWS 区域中的两个或更多桶之间完全同步。设置双向复制时，将创建从源桶（DOC-EXAMPLE-BUCKET-1）到包含副本的桶（DOC-EXAMPLE-BUCKET-2）的复制规则。然后，创建从包含副本的桶（DOC-EXAMPLE-BUCKET-2）到源桶（DOC-EXAMPLE-BUCKET-1）的第二个复制规则。

与所有复制规则一样，您可以将双向复制规则应用于整个 Amazon S3 桶，也可以应用于按前缀或对象标签筛选的对象子集。您还可以通过为每个复制规则启用副本修改同步来保持对于对象的元数据更改同步。您可以通过 Amazon S3 控制台、AWS CLI、AWS SDK、Amazon S3 REST API 或 AWS CloudFormation 启用副本修改同步。

要监控 Amazon CloudWatch 中对象和对象元数据的复制进度，请启用 S3 复制指标和通知。有关更多信息，请参阅使用复制指标和 Amazon S3 事件通知监控进度。

为多区域接入点创建双向复制规则

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
3. 请选择要更新的多区域接入点的名称。
4. 选择 Replication and failover（复制和失效转移）选项卡。
5. 向下滚动到 Replication rules（复制规则）部分，然后选择 Create replication rules（创建复制规则）。
6. 在 Create replication rules（创建复制规则）页面上，选择 Replicate objects among all specified buckets（在所有指定桶间复制对象）模板。Replicate objects among all specified buckets（在所有指定桶间复制对象）模板可为您的桶设置双向复制（具有失效转移功能）。

   > **Important**
   >
   > 当您使用此模板创建复制规则时，它们会替换已经分配给桶的任何现有复制规则。要添加或修改任何现有的复制规则而不是替换它们，请转到控制台中每个桶的 Management（管理）选项卡，然后在 Replication rules（复制规则）部分编辑规则。还可以使用 AWS CLI、AWS SDK 或 Amazon S3 REST API 添加或修改现有复制规则。有关更多信息，请参阅复制配置。

7. 在 Buckets（桶）部分中，选择至少两个要从中复制对象的桶。选择进行复制的所有桶都必须启用 S3 版本控制，并且每个桶必须位于不同的 AWS 区域中。有关 S3 版本控制的更多信息，请参阅在 Amazon S3 桶中使用版本控制。

   > **Note**
   >
   > 确保您具有建立复制所需的读取和复制权限，否则会遇到错误。有关更多信息，请参阅创建 IAM 角色。

8. 在 Replication rule configuration（复制规则配置）部分中，选择在创建复制规则时将其设置为 Enabled（启用）还是 Disabled（禁用）。

> **Note**
>
> 您无法在 Replication rule name（复制规则名称）框中输入名称。复制规则名称是根据您创建复制规则时的配置生成的。

9. 在 Scope（范围）部分中，为复制选择适当的范围。

   - 要复制整个桶，请选择 Apply to all objects in the bucket（应用到桶中的所有对象）。
   - 为了复制桶中对象的子集，请选择 Limit the scope of this rule using one or more filters（使用一个或多个筛选条件限制此规则的范围)。

     可按前缀、对象标签或二者的组合筛选对象。

     - 要将复制限制为名称以相同字符串（例如 `pictures`）开头的所有对象，请在 Prefix（前缀）框中输入前缀。

       如果您输入属于文件夹名称的前缀，则必须使用 `/`（正斜杠）作为最后一个字符（例如，`pictures/`）。
     - 要复制具有一个或多个对象标签的所有对象，请选择 Add tag（添加标签），然后在框中输入键值对。要添加另一个标签，请重复该过程。有关对象标签的更多信息，请参阅 使用标签对存储进行分类。

10. 向下滚动到 Additional replication options（其他复制选项）部分，然后选择要应用的复制选项。

> **Note**
>
> 我们建议您应用以下选项，尤其是在您打算将多区域接入点配置为支持失效转移时：
>
> - Replication time control（RTC）[复制时间控制（RTC）] – 要在可预测的时间范围内跨不同区域复制您的数据，您可以使用 S3 Replication Time Control（S3 RTC）。S3 RTC 在 15 分钟内复制 Amazon S3 中存储的 99.99% 的新对象（由服务等级协议提供支持）。有关更多信息，请参阅the section called “使用 S3 Replication Time Control”。
> - Replication metrics and notifications（复制指标和通知）– 启用 Amazon CloudWatch 指标以监控复制事件。
> - 删除标记复制 - 将复制由 S3 删除操作创建的删除标记。不会复制由生命周期规则创建的删除标记。有关更多信息，请参阅在桶之间复制删除标记。
> - Replica modification sync（副本修改同步）– 为每个复制规则启用副本修改同步，以使对于对象的元数据更改保持同步。有关更多信息，请参阅启用副本修改同步。

S3 RTC 和 CloudWatch 复制指标和通知需要额外收费。有关更多信息，请参阅 Amazon S3 定价和 Amazon CloudWatch 定价。

11. 如果您正在编写取代现有复制规则的新复制规则，请选择 I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten（我确认通过选择“创建复制规则”，这些现有的复制规则将被覆盖）。
12. 选择 Create replication rules（创建复制规则）以创建和保存新的双向复制规则。

查看多区域接入点的复制规则

使用多区域接入点，您可以设置单向复制规则或双向复制规则。有关如何管理复制规则的信息，请参阅使用 Amazon S3 控制台管理复制规则。

查看多区域接入点的复制规则

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Multi-Region Access Points（多区域接入点）。
3. 请选择多区域接入点的名称。
4. 选择 Replication and failover（复制和失效转移）选项卡。
5. 向下滚动到复制规则部分。本节将列出已为多区域接入点创建的所有复制规则。

> **Note**
>
> 如果您已将来自其他账户的桶添加到此多区域接入点，则必须从桶拥有者获得 `s3:GetBucketReplication` 权限才能查看该桶的复制规则。

将多区域接入点与支持的 API 操作结合使用

Amazon S3 提供了一组操作用来管理多区域访问点。Amazon S3 同步处理其中一些操作，异步处理另一些操作。当您调用异步操作时，Amazon S3 首先同步授权请求的操作。如果授权成功，Amazon S3 将返回一个令牌，您可以使用该令牌来跟踪所请求操作的进度和结果。

> **Note**
>
> 通过 Amazon S3 控制台发出的请求始终是同步的。控制台等待，直到请求完成，然后再允许您提交另一个请求。

您可以使用控制台查看异步操作的当前状态和结果，也可以使用 AWS CLI、AWS SDK 或 REST API 中的 `DescribeMultiRegionAccessPointOperation`。Amazon S3 在响应异步操作时提供跟踪令牌。您将该跟踪令牌作为 `DescribeMultiRegionAccessPointOperation`。当您包括跟踪令牌时，Amazon S3 会返回指定操作的当前状态和结果，包括任何错误或相关资源信息。Amazon S3 执行 `DescribeMultiRegionAccessPointOperation` 同步操作。

所有创建或维护多区域接入点的控制面板请求都必须路由到 US West (Oregon) 区域。对于多区域接入点数据面板请求，无需指定区域。对于多区域接入点失效转移控制面板，请求必须路由到五个支持的区域之一。有关支持多区域接入点的区域的更多信息，请参阅多区域接入点限制和限制。

此外，您必须将 `s3:ListAllMyBuckets` 权限授予请求管理多区域接入点的用户、角色或其他 AWS Identity and Access Management（IAM）实体。

以下示例演示如何在 Amazon S3 中将多区域接入点与兼容操作结合使用。

主题

- 多区域接入点与 AWS 服务和 AWS SDK 的兼容性
- 多区域接入点与 S3 操作的兼容性
- 查看您的多区域接入点路由配置
- 更新您的底层 Amazon S3 存储桶策略
- 更新多区域接入点路由配置
- 将对象添加到多区域接入点的桶中
- 从多区域接入点检索对象
- 列出存储在多区域接入点底层桶中的对象
- 将预签名 URL 与多区域接入点结合使用
- 将配置了申请方付款的桶与多区域接入点结合使用

多区域接入点与 AWS 服务和 AWS SDK 的兼容性

要在需要 Amazon S3 存储桶名称的应用程序中使用多区域接入点，请在使用 AWS SDK 发出请求时使用多区域接入点的 Amazon 资源名称（ARN）。要检查哪些 AWS SDK 与多区域接入点兼容，请参阅与 AWS SDK 的兼容性。

多区域接入点与 S3 操作的兼容性

您可以使用以下 Amazon S3 数据面板 API 操作，对桶中与您的多区域接入点关联的对象执行操作。以下 S3 操作可以接受多区域接入点 ARN：

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetObject
- GetObjectAcl
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadObject
- ListMultipartUploads
- ListObjectsV2
- ListParts
- PutObject
- PutObjectAcl
- PutObjectLegalHold
- PutObjectRetention
- PutObjectTagging
- RestoreObject
- UploadPart

> **Note**
>
> 当使用多区域接入点 ARN 时，多区域接入点仅支持使用多区域接入点的复制操作作为目标。

您可以使用以下 Amazon S3 控制面板操作来创建和管理多区域接入点：

- CreateMultiRegionAccessPoint
- DescribeMultiRegionAccessPointOperation
- GetMultiRegionAccessPoint
- GetMultiRegionAccessPointPolicy
- GetMultiRegionAccessPointPolicyStatus
- GetMultiRegionAccessPointRoutes
- ListMultiRegionAccessPoints
- PutMultiRegionAccessPointPolicy
- SubmitMultiRegionAccessPointRoutes

查看您的多区域接入点路由配置

AWS CLI

以下示例命令检索您的多区域接入点路由配置，以便您可以查看桶的当前路由状态。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
```

SDK for Java

以下 SDK for Java 代码检索您的多区域接入点路由配置，以便您可以查看桶的当前路由状态。要使用此示例语法，请将 *user input placeholders* 替换为您自己的信息。

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(credentialsProvider)
```

```
    .build();

GetMultiRegionAccessPointRoutesRequest request =
 GetMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .build();

GetMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.getMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

以下 SDK for JavaScript 代码检索您的多区域接入点路由配置，以便您可以查看桶的当前路由状态。要使用此示例语法，请将 *user input placeholders* 替换为您自己的信息。

```
const REGION = 'us-east-1'

const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new GetMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
      })
    )
    console.log('Success', data)
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python

以下 SDK for Python 代码检索您的多区域接入点路由配置，以便您可以查看桶的当前路由状态。要使用此示例语法，请将 *user input placeholders* 替换为您自己的信息。

```
s3.get_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap)['Routes']
```

更新您的底层 Amazon S3 存储桶策略

要授予适当的访问权限，您还必须更新底层 Amazon S3 存储桶策略。以下示例将访问控制委托给多区域接入点策略。在将访问控制委托给多区域接入点策略后，当通过多区域接入点发出请求时，桶策略不再用于访问控制。

以下是将访问控制委托给多区域接入点策略的示例桶策略。要使用这一示例桶策略，请将 *user input placeholders* 替换为您自己的信息。要通过 AWS CLI `put-bucket-policy` 命令应用此策略，如下一个示例所示，请将策略保存在文件中，例如 `policy.json`。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": { "AWS": "*" },
    "Effect": "Allow",
    "Action": ["s3:*"],
    "Resource": ["arn:aws:s3:::111122223333/*", "arn:aws:s3:::amzn-s3-demo-bucket"],
    "Condition": {
      "StringEquals": {
        "s3:DataAccessPointAccount": "444455556666"
      }
    }
  }
}
```

以下 `put-bucket-policy` 示例命令将更新后的 S3 存储桶策略与您的 S3 存储桶关联起来：

```
aws s3api put-bucket-policy
  --bucket amzn-s3-demo-bucket
  --policy file:///tmp/policy.json
```

更新多区域接入点路由配置

以下示例命令更新多区域接入点路由配置。可以对以下五个区域运行多区域接入点路由命令：

- `ap-southeast-2`

- ap-northeast-1
- us-east-1
- us-west-2
- eu-west-1

在多区域接入点路由配置中，您可以将桶设置为主动或被动路由状态。主动桶接收流量，而被动桶不接收流量。您可以通过将桶的 TrafficDialPercentage 值设置为 100（表示主动）或 0（表示被动）来设置桶的路由状态。

AWS CLI

以下示例命令将更新您的多区域接入点路由配置。在此示例中，*amzn-s3-demo-bucket1* 设置为主动状态，*amzn-s3-demo-bucket2* 设置为被动状态。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 111122223333
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
--route-updates Bucket=amzn-s3-demo-bucket1,TrafficDialPercentage=100
                Bucket=amzn-s3-demo-bucket2,TrafficDialPercentage=0
```

SDK for Java

以下 SDK for Java 代码将更新您的多区域接入点路由配置。要使用此示例语法，请将 *user input placeholders* 替换为您自己的信息。

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.ap-southeast-2)
    .credentialsProvider(credentialsProvider)
    .build();

SubmitMultiRegionAccessPointRoutesRequest request =
 SubmitMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .routeUpdates(
        MultiRegionAccessPointRoute.builder()
            .region("eu-west-1")
            .trafficDialPercentage(100)
```

```
            .build(),
        MultiRegionAccessPointRoute.builder()
            .region("ca-central-1")
            .bucket("111122223333")
            .trafficDialPercentage(0)
            .build()
    )
    .build();

SubmitMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.submitMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

以下 SDK for JavaScript 代码将更新您的多区域接入点路由配置。要使用此示例语法，请将 *user input placeholders* 替换为您自己的信息。

```
const REGION = 'ap-southeast-2'

const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new SubmitMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
        RouteUpdates: [
          {
            Region: 'eu-west-1',
            TrafficDialPercentage: 100,
          },
          {
            Region: 'ca-central-1',
            Bucket: 'amzn-s3-demo-bucket1',
            TrafficDialPercentage: 0,
          },
        ],
      })
    )
    console.log('Success', data)
```

```
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python

以下 SDK for Python 代码将更新您的多区域接入点路由配置。要使用此示例语法，请将 *user input placeholders* 替换为您自己的信息。

```
s3.submit_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap,
        RouteUpdates= [{
            'Bucket': DOC-EXAMPLE-BUCKET,
            'Region': ap-southeast-2,
            'TrafficDialPercentage': 10
        }])
```

将对象添加到多区域接入点的桶中

要将对象添加到与多区域接入点关联的桶中，您可以使用 PutObject 操作。要使多区域接入点中的所有桶保持同步，请启用跨区域复制。

> **Note**
>
> 要使用此操作，您必须对多区域接入点拥有 `s3:PutObject` 权限。有关多区域接入点权限要求的更多信息，请参阅权限。

AWS CLI

以下示例数据面板请求将 *example.txt* 上传到指定的多区域接入点。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3api put-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --key example.txt --
body example.txt
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
        .key("example.txt")
        .build();

s3Client.putObject(objectRequest, RequestBody.fromString("Hello S3!"));
```

SDK for JavaScript

```
const client = new S3Client({});

async function putObjectExample() {
    const command = new PutObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt",
        Body: "Hello S3!",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.put_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
```

```
    Key='example.txt',
    Body='Hello S3!'
)
```

从多区域接入点检索对象

要从多区域接入点检索对象，可以使用 GetObject 操作。

> **Note**
>
> 要使用此 API 操作，您必须对多区域接入点拥有 s3:GetObject 权限。有关多区域接入点权限要求的更多信息，请参阅权限。

AWS CLI

以下示例数据面板请求从指定的多区域接入点检索 *example.txt* 并将其下载为 *downloaded_example.txt*。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3api get-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --
key example.txt downloaded_example.txt
```

SDK for Java

```
S3Client s3 = S3Client
   .builder()
   .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example.txt")
    .build();

s3Client.getObject(getObjectRequest);
```

SDK for JavaScript

```
const client = new S3Client({})
```

```
async function getObjectExample() {
    const command = new GetObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt"
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3

client = boto3.client('s3')
client.get_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt'
)
```

列出存储在多区域接入点底层桶中的对象

要返回存储在多区域接入点底层桶中的对象的列表，请使用 ListObjectsV2 操作。在以下示例命令中，使用多区域接入点的 ARN 列出了指定多区域接入点的所有对象。在这种情况下，多区域接入点 ARN 为：

`arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap`

> **Note**
>
> 要使用此 API 操作，您必须对于多区域接入点和底层桶拥有 `s3:ListBucket` 权限。有关多区域接入点权限要求的更多信息，请参阅权限。

AWS CLI

以下示例数据面板请求列出了桶中的对象，该桶位于 ARN 指定的多区域接入点的底层。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3api list-objects-v2 --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

String bucketName = "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap";

ListObjectsV2Request listObjectsRequest = ListObjectsV2Request
    .builder()
    .bucket(bucketName)
    .build();

 s3Client.listObjectsV2(listObjectsRequest);
```

SDK for JavaScript

```
const client = new S3Client({});

async function listObjectsExample() {
    const command = new ListObjectsV2Command({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python

```
import boto3
```

```
client = boto3.client('s3')
client.list_objects_v2(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap'
)
```

将预签名 URL 与多区域接入点结合使用

您可以使用预签名 URL 生成一个 URL，该 URL 允许其他人通过 Amazon S3 多区域接入点访问 Amazon S3 存储桶。创建预签名 URL 时，请将它与特定的对象操作相关联，例如 S3 上传（`PutObject`）或 S3 下载（`GetObject`）。您可以共享预签名 URL，任何有权访问该 URL 的人都可以像原始签名用户一样执行嵌入在 URL 中的操作。

预签名 URL 具有到期日期。到达到期时间后，URL 将不再起作用。

在将 S3 多区域接入点与预签名 URL 结合使用之前，请检查 AWS SDK 与 SigV4A 算法的兼容性。验证您的 SDK 版本是否支持将 SigV4A 作为用于签署全局 AWS 区域请求的签名实现。有关在 Amazon S3 中使用预签名 URL 的更多信息，请参阅使用预签名 URL 共享对象。

以下示例说明如何将多区域接入点与预签名 URL 结合使用。要使用这些示例，请将 *user input placeholders* 替换为您自己的信息。

AWS CLI

```
aws s3 presign
 arn:aws:s3::123456789012:accesspoint/MultiRegionAccessPoint_alias/example-file.txt
```

SDK for Python

```
import logging
import boto3
from botocore.exceptions import ClientError

s3_client = boto3.client('s3',aws_access_key_id='xxx',aws_secret_access_key='xxx')
s3_client.generate_presigned_url(HttpMethod='PUT',ClientMethod="put_object",
 Params={'Bucket':'arn:aws:s3::123456789012:accesspoint/
abcdef0123456.mrap','Key':'example-file'})
```

SDK for Java

```
S3Presigner s3Presigner = S3Presigner.builder()
```

```
    .credentialsProvider(StsAssumeRoleCredentialsProvider.builder()
        .refreshRequest(assumeRole)
        .stsClient(stsClient)
        .build())
    .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example-file")
    .build();

GetObjectPresignRequest preSignedReq = GetObjectPresignRequest.builder()
    .getObjectRequest(getObjectRequest)
    .signatureDuration(Duration.ofMinutes(10))
    .build();

PresignedGetObjectRequest presignedGetObjectRequest =
 s3Presigner.presignGetObject(preSignedReq);
```

> **Note**
>
> 要将 SigV4A 与临时安全凭证结合使用 [例如，在使用 IAM 角色时]，请确保从 AWS Security Token Service（AWS STS）中的区域端点而非全局端点请求临时凭证。如果您对 AWS STS（`sts.amazonaws.com`）使用全局端点，则 AWS STS 将从全局端点生成临时证书，而 Sig4A 不支持全局端点。因此，您将得到错误。要解决此问题，请使用列出的任何适用于 AWS STS 的区域性端点。

将配置了申请方付款的桶与多区域接入点结合使用

如果与您的多区域接入点关联的 S3 存储桶配置为使用申请方付款，则申请方将支付桶请求费用、下载费用以及任何与多区域接入点相关的费用。有关更多信息，请参阅 Amazon S3 定价。

以下是向连接到申请方付款桶的多区域接入点发出数据面板请求的示例。

AWS CLI

要从连接到申请方付款桶的多区域接入点下载对象，您必须指定 `--request-payer requester` 作为 get-object 请求的一部分。您还必须指定桶中文件的名称以及应存储所下载文件的位置。

```
aws s3api get-object --bucket MultiRegionAccessPoint_ARN --request-payer requester
 --key example-file-in-bucket.txt example-location-of-downloaded-file.txt
```

SDK for Java

要从连接到申请方付款桶的多区域接入点下载对象，您必须指定 RequestPayer.REQUESTER 作为 GetObject 请求的一部分。您还必须指定桶中文件的名称及其存储位置。

```
GetObjectResponse getObjectResponse = s3Client.getObject(GetObjectRequest.builder()
    .key("example-file.txt")
    .bucket("arn:aws:s3::
123456789012:accesspoint/abcdef0123456.mrap")
    .requestPayer(RequestPayer.REQUESTER)
    .build()
).response();
```

监视和记录通过多区域接入点发出的对底层资源的请求

Amazon S3 记录通过多区域接入点发出的请求以及对管理这些接入点的 API 操作（如 CreateMultiRegionAccessPoint 和 GetMultiRegionAccessPointPolicy）发出的请求。通过接入点向 Amazon S3 发出的请求将显示在 S3 服务器访问日志和具有多区域接入点主机名的 AWS CloudTrail 日志中。接入点的主机名采用此形式：*MRAP_alias*.accesspoint.s3-global.amazonaws.com。例如，假设您具有以下桶和接入点配置：

- 区域 us-west-2 中包含对象 my-image.jpg 且名为 my-bucket-usw2 的桶。
- 区域 ap-south-1 中包含对象 my-image.jpg 且名为 my-bucket-aps1 的桶。
- 区域 eu-central-1 中不包含名为 my-image.jpg 的对象且名为 my-bucket-euc1 的桶。
- 名为 my-mrap 别名 mfzwi23gnjvgw.mrap 的多区域接入点配置为满足来自所有三个桶的请求。
- 您的 AWS 账户 ID 是 123456789012。

为了直接通过日志中显示的主机名为 *bucket_name*.s3.*Region*.amazonaws.com 的桶检索 my-image.jpg 而发出的请求。

如果您改为通过多区域接入点发出请求，Amazon S3 首先确定不同区域中的哪个桶距离最近。Amazon S3 确定用于满足请求的桶后，它会将请求发送到该桶并使用多区域接入点主机名记录操作。在此示例中，如果 Amazon S3 将请求中继到 my-bucket-aps1，则您的

日志将反映对 my-bucket-aps1 中 my-image.jpg 的成功 GET 请求，使用的主机名为 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com。

> **Important**
>
> 多区域接入点不了解底层桶的数据内容。因此，收到请求的桶可能不包含所请求的数据。例如，如果 Amazon S3 确定 my-bucket-euc1 桶距离最近，则您的日志将反映对于 my-bucket-euc1 中 my-image.jpg 的失败 GET 请求，使用的主机名为 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com。如果请求已改而路由到 my-bucket-usw2，您的日志将表明成功的 GET 请求。

有关 Amazon S3 服务器访问日志的更多信息，请参阅 使用服务器访问日志记录来记录请求。有关 AWS CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》中的什么是 AWS CloudTrail？。

监控和记录对多区域接入点管理 API 操作发出的请求

Amazon S3 提供了多个 API 操作来管理多区域接入点，例如 CreateMultiRegionAccessPoint 和 GetMultiRegionAccessPointPolicy。当您使用 AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API 请求这些 API 操作时，Amazon S3 将异步处理这些请求。只要您对请求具有相应的权限，Amazon S3 将返回这些请求的令牌。您可以将此令牌与 DescribeAsyncOperation 用来帮助您查看正在进行的异步操作的状态。Amazon S3 同步处理 DescribeAsyncOperation 请求。要查看异步请求的状态，可以使用 Amazon S3 控制台、AWS CLI、SDK 或 REST API。

> **Note**
>
> 控制台仅显示前 14 天内发出的异步请求的状态。要查看旧请求的状态，请使用 AWS CLI、开发工具包或 REST API。

异步管理操作可以处于以下几种状态之一：

NEW

Amazon S3 已收到请求，并准备执行该操作。

IN_PROGRESS

Amazon S3 当前正在执行该操作。

SUCCESS

操作已成功。响应包括相关信息，例如 CreateMultiRegionAccessPoint 请求。

FAILED

该操作失败。响应包含一条错误消息，指示请求失败的原因。

将 AWS CloudTrail 与多区域接入点结合使用

您可以使用 AWS CloudTrail 来查看、搜索、下载、归档、分析和响应您的 AWS 基础设施中的账户活动。通过多区域接入点和 CloudTrail 日志记录，您可以确定以下各项：

- 谁或什么执行了哪个操作
- 对哪些资源执行了操作
- 事件发生的时间
- 有关事件的其他详细信息

您可以使用此日志记录信息来帮助您分析和响应通过多区域接入点发生的活动。

如何设置 AWS CloudTrail 多区域接入点

要针对与创建或维护多区域接入点相关的任何操作启用 CloudTrail 日志记录，您必须配置 CloudTrail 日志记录来记录美国西部（俄勒冈州）区域中的事件。您必须以这种方式设置日志记录配置，而无论您在发出请求时所在的区域，也无论多区域接入点支持哪些区域。创建或维护多区域接入点的所有请求都通过美国西部（俄勒冈州）区域路由。我们建议您将此区域添加到现有跟踪中，或者创建包含此区域以及与多区域接入点关联的所有区域的新跟踪。

Amazon S3 记录通过多区域接入点发出的所有请求以及对管理接入点的 API 操作（如 CreateMultiRegionAccessPoint 和 GetMultiRegionAccessPointPolicy）发出的请求。当您通过多区域接入点记录这些请求时，它们将显示在 AWS CloudTrail 日志中包含多区域接入点的主机名。例如，如果您通过带有别名 mfzwi23gnjvgw.mrap 的多区域接入点向桶发出请求，则 CloudTrail 日志中的条目的主机名将为 mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com。

多区域接入点将请求路由到最近的桶。由于这种行为，当您查看多区域接入点的 CloudTrail 日志时，您将看到对底层桶发出的请求。其中一些请求可能是对桶的直接请求，这些请求不是通过多区域接入点路由的。在查看流量时，请记住这一事实。当桶位于多区域接入点中时，仍可直接向该桶发出请求，而无需通过多区域接入点。

创建和管理多区域接入点涉及异步事件。异步请求在 CloudTrail 日志中没有完成事件。有关异步请求的更多信息，请参阅 监控和记录对多区域接入点管理 API 操作发出的请求。

有关 AWS CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》中的什么是 AWS CloudTrail？。

Amazon S3 安全性

AWS 十分重视云安全性。作为 AWS 客户，您将从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

安全性是 AWS 和您的共同责任。责任共担模式将其描述为云本身的安全性和云中的安全性：

云的安全性

AWS 负责保护在 AWS Cloud 中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为 AWS 合规性计划的一部分，我们的安全措施的有效性定期由第三方审计员进行测试和验证。要了解适用于 Amazon S3 的合规性计划，请参阅AWS合规性计划范围内的 服务。

云中的安全性

您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您组织的要求以及适用的法律法规。对于 Amazon S3，您的责任包括以下各个方面：

- 管理数据，包括对象所有权和加密。
- 对资产进行分类。
- 使用 IAM 角色管理您数据的访问权以及其他服务配置来应用适当的权限。
- 启用侦测性控制，例如适用于 Amazon S3 的 AWS CloudTrail 或 Amazon GuardDuty。

此文档将帮助您了解如何在使用 Amazon S3 时应用责任共担模式。以下主题说明如何配置 Amazon S3 以实现您的安全性和合规性目标。您还将了解如何使用其他 AWS 服务来帮助您监控和保护您的 Amazon S3 资源。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

主题

- Amazon S3 中的数据保护
- 利用加密来保护数据

- 互联网络流量隐私保护
- AWS PrivateLink：表示 Amazon S3
- 访问控制
- Amazon S3 中的日志记录和监控
- Amazon S3 的合规性验证
- Amazon S3 中的恢复能力
- Amazon S3 中的基础设施安全性
- Amazon S3 中的配置和漏洞分析
- Amazon S3 的安全最佳实践
- 使用托管式 AWS 安全服务监控数据安全

Amazon S3 中的数据保护

Amazon S3 为任务关键型和主要数据存储提供了高度持久的存储基础设施。S3 Standard、S3 Intelligent-Tiering、S3 Standard-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive 在 AWS 区域 中至少三个可用区的多个设备上冗余存储对象。可用区是 AWS 区域 中一个或多个具有冗余电源、网络和连接的离散数据中心。可用区与任何其他可用区之间有意义的距离（许多公里数）在物理上隔开，尽管所有可用区之间的距离都在 100 公里（60 英里）以内。S3 One Zone-IA 存储类将数据冗余存储在单个可用区中的多个设备上。这些服务旨在通过快速检测和修复任何丢失的冗余来处理并发设备故障，还可以使用校验和定期验证数据的完整性。

Amazon S3 standard 存储提供以下特征：

- 以 Amazon S3 服务等级协议作为后盾
- 设计目的是在指定年度内为对象提供 99.999999999% 的持久性和 99.99% 的可用性
- S3 Standard、S3 Intelligent-Tiering、S3 Standard-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 都是为了在整个 Amazon S3 可用区丢失的情况下维持数据。

Amazon S3 使用版本控制功能进一步保护您的数据。对于 Amazon S3 存储桶中存储的每个对象，您可以使用版本控制功能来保存、检索和还原它们的每个版本。使用版本控制能够轻松从用户意外操作和应用程序故障中恢复数据。默认情况下，请求会检索最新写入的版本。您可以通过指定请求中对象的版本，来检索该对象的旧版本。

除了 S3 版本控制之外，您还可以使用 Amazon S3 对象锁定和 S3 复制来保护您的数据。有关更多信息，请参阅教程：使用 S3 版本控制、S3 对象锁定和 S3 复制保护 Amazon S3 上的数据免遭意外删除或应用程序错误。

出于数据保护的目的，我们建议您保护 AWS 账户凭证并使用 AWS Identity and Access Management 设置单个用户账户，以便仅向每个用户提供履行其工作职责所需的权限。

如果在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

下面的安全最佳实践也探讨 Amazon S3 中的数据保护：

- Implement server-side encryption
- Enforce encryption of data in transit
- Consider using Macie with Amazon S3
- Identify and audit all your Amazon S3 buckets
- Monitor Amazon Web Services security advisories

利用加密来保护数据

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

数据保护指在数据传输（发往和离开 Amazon S3 时）和处于静态（存储在 Amazon S3 数据中心的磁盘上时）期间保护数据。您可以使用安全套接字层/传输层安全性（SSL/TLS）或客户端加密来保护传输中数据。为保护 Amazon S3 中的静态数据，您具有以下选项：

- 服务器端加密 – Amazon S3 在将对象保存到 AWS 数据中心的磁盘之前对其进行加密，然后在下载对象时对其进行解密。

默认情况下，所有 Amazon S3 存储桶都配置了加密，所有上传到 S3 存储桶的新对象都会自动静态加密。具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）是 Amazon S3 中每个存储桶的默认加密配置。要使用其他类型的加密，您可以指定要在 S3 PUT 请求中使用的服务器端加密类型，也可以在目标存储桶中设置默认加密配置。

如果您想在 PUT 请求中指定不同的加密类型，则可以使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）或具有客户提供的密钥的服务器端加密（SSE-C）。如果您想在目标存储桶中设置不同的默认加密配置，则可以使用 SSE-KMS 或 DSSE-KMS。

有关服务器端加密的每个选项的更多信息，请参阅使用服务器端加密保护数据。

要配置服务器端加密，请参阅：

- 指定具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）
- 使用 AWS KMS (SSE-KMS) 指定服务器端加密
- the section called “指定 DSSE-KMS”
- 指定使用客户提供的密钥的服务器端加密 (SSE-C)。

- 客户端加密 – 您在客户端加密数据并将加密的数据上传到 Amazon S3。在这种情况下，您需要管理加密过程、加密密钥和相关的工具。

  要配置客户端加密，请参阅 使用客户端加密保护数据。

要查看存储字节的加密百分比，您可以使用 Amazon S3 Storage Lens 存储统计管理工具指标。S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。有关更多信息，请参阅使用 S3 Storage Lens 存储统计管理工具访问存储活动和使用情况。有关指标的完整列表，请参阅 S3 Storage Lens 存储统计管理工具指标词汇表。

有关服务器端加密和客户端加密的更多信息，请查看下面的主题。

主题

- 使用服务器端加密保护数据
- 使用客户端加密保护数据

使用服务器端加密保护数据

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

服务器端加密是指由接收数据的应用程序或服务在目标位置对数据进行加密。Amazon S3 在将您的数据写入 AWS 数据中心内的磁盘时会在对象级别加密这些数据，并在您访问这些数据时解密这些数据。只要您验证了您的请求并且拥有访问权限，您访问加密和未加密对象的方式就没有区别。例如，如果您使用预签名的 URL 来共享您的对象，那么对于加密和解密的对象，该 URL 的工作方式是相同的。此外，在您列出存储桶中的对象时，列表 API 操作会返回所有对象的列表（无论对象是否加密）。

默认情况下，所有 Amazon S3 存储桶都配置了加密，所有上传到 S3 存储桶的新对象都会自动静态加密。具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）是 Amazon S3 中每个存储桶的默认加密配置。要使用其他类型的加密，您可以指定要在 S3 PUT 请求中使用的服务器端加密类型，也可以在目标存储桶中设置默认加密配置。

如果您想在 PUT 请求中指定不同的加密类型，则可以使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）或具有客户提供的密钥的服务器端加密（SSE-C）。如果您想在目标存储桶中设置不同的默认加密配置，则可以使用 SSE-KMS 或 DSSE-KMS。

> **Note**
>
> 您不能对同一个对象应用不同类型的服务器端加密。

如果您需要加密现有对象，请使用 S3 批量操作和 S3 清单。有关更多信息，请参阅使用 Amazon S3 批量操作加密对象和对 Amazon S3 对象执行大规模批量操作。

对于服务器端加密，您有四个互斥的选项，具体取决于您选择如何管理加密密钥和要应用的加密层数。

具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）

默认情况下，所有 Amazon S3 存储桶都配置了加密。服务器端加密的默认选项是使用 Amazon S3 托管式密钥（SSE-S3）。每个对象都使用唯一的密钥来进行加密。作为额外的保护措施，SSE-S3 使用定期轮换的根密钥加密密钥本身。SSE-S3 使用可用的最强数据块密码之一 [即 256 位高级加密标准（AES-256）] 来加密您的数据。有关更多信息，请参阅 使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）

具有 AWS KMS keys 的服务器端加密（SSE-KMS）是通过将 AWS KMS 服务与 Amazon S3 集成来提供的。使用 AWS KMS，您可以更好地控制您的密钥。例如，您可以查看单独的密钥、编辑控制策略以及遵循 AWS CloudTrail 中的密钥。此外，您还可以创建和管理客户自主管理型密钥，或者使用对于您、服务和区域为唯一的 AWS 托管式密钥。有关更多信息，请参阅 使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。

具有 AWS Key Management Service（AWS KMS）密钥的双层服务器端加密（DSSE-KMS）

使用 AWS KMS keys 的双层服务器端加密（DSSE-KMS）与 SSE-KMS 类似，但 DSSE-KMS 应用的是两层单独的对象级加密，而不是一层。由于两层加密都应用于服务器端的对象，因此您可以使用各种 AWS 服务和工具来分析 S3 中的数据，同时使用可以满足合规性要求的加密方法。有关更多信息，请参阅 使用具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）。

具有客户提供密钥的服务器端加密（SSE-C）

使用具有客户提供的密钥的服务器端加密（SSE-C）时，您管理加密密钥，而 Amazon S3 管理加密（在它对磁盘进行写入时）和解密（在您访问您的对象时）。有关更多信息，请参阅 使用具有客户提供的密钥的服务器端加密（SSE-C）。

Amazon S3 现在会自动加密所有新对象

Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。使用 256 位高级加密标准（AES-256）的 SSE-S3 会自动应用于所有新存储桶和任何尚未配置默认加密的现有 S3 存储桶。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface（AWS CLI）和 AWS SDK 中的附加 Amazon S3 API 响应标头。

以下各部分回答了有关此更新的问题。

Amazon S3 会更改已配置了默认加密的现有存储桶的默认加密设置吗？

不会。对于已配置 SSE-S3 或具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）的现有存储桶的默认加密配置，不会发生任何更改。有关如何设置存储桶的默认加密行为的更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。有关 SSE-S3 和 SSE-KMS 加密设置的更多信息，请参阅使用服务器端加密保护数据。

是否会在未配置默认加密的现有存储桶上启用默认加密？

是。Amazon S3 现在在所有现有未加密的存储桶上配置默认加密，以应用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3），作为对上传到这些存储桶的新对象的基本加密级别。已在现有未加密存储桶中的对象不会自动加密。

如何查看新对象上传的默认加密状态？

目前，您可以在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中查看新上传的对象的默认加密状态，并可将其作为 AWS Command Line Interface（AWS CLI）和 AWS SDK 中的附加 Amazon S3 API 响应标头。

- 要查看您的 CloudTrail 事件，请参阅《AWS CloudTrail 用户指南》中的在 CloudTrail 控制台中查看 CloudTrail 事件。CloudTrail 日志为针对 Amazon S3 的 PUT 和 POST 请求提供 API 跟踪。当使用默认加密来加密存储桶中的对象时，关于 PUT 和 POST API 请求的 CloudTrail 日志将包含以下字段作为名称/值对：`"SSEApplied":"Default_SSE_S3"`。
- 要查看 S3 清单中新对象上传的自动加密状态，请将 S3 清单报告配置为包含 Encryption（加密）元数据字段，然后在报告中查看每个新对象的加密状态。有关更多信息，请参阅设置 Amazon S3 清单。
- 要在 S3 Storage Lens 存储统计管理工具中查看新对象上传的自动加密状态，请配置 S3 Storage Lens 存储统计管理工具控制面板，并在控制面板的 Data protection（数据保护）类别中查看 Encrypted bytes（加密字节数）和 Encrypted object count（加密对象计数）指标。有关更多信息，请参阅创建 Amazon S3 Storage Lens 存储统计管理工具控制面板 和在控制面板上查看 S3 Storage Lens 存储统计管理工具指标。
- 要在 Amazon S3 控制台中查看存储桶级别的自动加密状态，请在 Amazon S3 控制台中查看 Amazon S3 存储桶的默认加密。有关更多信息，请参阅 配置默认加密。
- 要在 AWS Command Line Interface（AWS CLI）和 AWS SDK 中将自动加密状态作为附加 Amazon S3 API 响应标头来查看，请在使用对象操作 API（如 PutObject 和 GetObject）时检查响应标头 `x-amz-server-side-encryption`。

我需要做什么才能利用这一更改？

您无需对现有应用程序进行任何更改。由于针对您的所有存储桶都启用了默认加密，因此上传到 Amazon S3 的所有新对象都会自动加密。

我能否对正写入我的存储桶的新对象禁用加密？

不能。SSE-S3 是新的基本加密级别，适用于上传到您的存储桶的所有新对象。您无法再为新对象上传禁用加密。

我的费用会受到影响吗？

不会。使用 SSE-S3 进行默认加密不会产生额外的成本。像往常一样，将向您收取存储、请求和其他 S3 特征的费用。有关定价信息，请参阅 Amazon S3 定价。

Amazon S3 会加密我现有的未加密对象吗？

不会。从 2023 年 1 月 5 日开始，Amazon S3 仅自动加密新上传的对象。要加密现有对象，您可以使用 S3 批量操作创建对象的加密副本。这些加密副本将保留现有的对象数据和名称，并将使用您指定的加密密钥进行加密。有关更多详细信息，请参阅 AWS 存储博客中的 Encrypting objects with Amazon S3 Batch Operations（使用 Amazon S3 批量操作加密对象）。

在此版本之前，我没有为我的存储桶启用加密。我需要更改访问对象的方式吗？

不需要。使用 SSE-S3 进行默认加密，在将您的数据写入 Amazon S3 时会自动加密这些数据，并在您访问时解密这些数据。访问自动加密的对象的方式没有变化。

我需要更改访问客户端加密的对象的方式吗？

不需要。所有在上传到 Amazon S3 之前经过加密的客户端加密对象都作为 Amazon S3 中已加密的加密文字对象到达。这些对象现在将具有额外的 SSE-S3 加密层。使用客户端加密对象的工作负载不要求对客户端服务或授权设置进行任何更改。

> **Note**
>
> 未使用 AWS 提供者的已更新版本的 HashiCorp Terraform 用户在创建没有客户定义的加密配置的新 S3 存储桶后，可能会看到意想不到的偏差。为避免这种偏差，请将您的 Terraform AWS 提供者版本更新为以下版本之一：任何 4.x 发行版、3.76.1 或 2.70.4。

使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

默认情况下，向 Amazon S3 存储桶上传的所有新对象都使用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）进行加密。

服务器端加密可保护静态数据。Amazon S3 使用唯一的密钥来加密每个对象。作为额外的保护，它将使用定期轮换的密钥加密密钥本身。Amazon S3 服务器端加密使用 256 位高级加密标准 Galois/Counter 模式（AES-GCM）对所有上传的对象进行加密。

使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）不会产生额外费用。然而，请求配置默认加密特征会产生标准 Amazon S3 请求费用。有关定价的信息，请参阅 Amazon S3 定价。

如果您要求仅使用 Amazon S3 托管密钥对上传的数据进行加密，则可以使用以下存储桶策略。例如，以下存储桶策略拒绝上传对象的权限，除非请求包含用于请求服务器端加密的 `x-amz-server-side-encryption` 标头：

```
{
  "Version": "2012-10-17",
  "Id": "PutObjectPolicy",
  "Statement": [
    {
      "Sid": "DenyObjectsThatAreNotSSES3",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
```

```
      }
    }
  ]
}
```

> **Note**
>
> 服务器端加密仅加密对象数据而非加密对象元数据。

服务器端加密的 API 支持

默认情况下，所有 Amazon S3 存储桶都配置了加密，所有上传到 S3 存储桶的新对象都会自动静态加密。具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）是 Amazon S3 中每个存储桶的默认加密配置。要使用其他类型的加密，您可以指定要在 S3 PUT 请求中使用的服务器端加密类型，也可以在目标存储桶中设置默认加密配置。

如果您想在 PUT 请求中指定不同的加密类型，则可以使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）或具有客户提供的密钥的服务器端加密（SSE-C）。如果您想在目标存储桶中设置不同的默认加密配置，则可以使用 SSE-KMS 或 DSSE-KMS。

要使用对象创建 REST API 配置服务器端加密，必须提供 `x-amz-server-side-encryption` 请求标头。有关 REST API 的信息，请参阅 使用 REST API。

以下 Amazon S3 API 支持此标头：

- PUT 操作 - 在使用 PUT API 上传数据时指定请求标头。有关更多信息，请参阅 PUT Object。
- 启动分段上传 - 当使用分段上传 API 操作上传大型对象时，在启动请求中指定标头。有关更多信息，请参阅启动分段上传。
- COPY 操作 - 在复制对象时，您同时具有源对象和目标对象。有关更多信息，请参阅 PUT Object - 复制。

> **Note**
>
> 当使用 POST 操作上传对象时，请在表单字段中提供相同的信息，而不是提供请求标头。有关更多信息，请参阅 POST 对象。

AWS SDK 还提供了一个可用于请求服务器端加密的包装程序 API。您还可以使用 AWS Management Console 来上传对象并请求服务器端加密。

有关更多常规信息，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS 概念。

主题

- 指定具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）

指定具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

默认情况下，所有 Amazon S3 存储桶都配置了加密，所有上传到 S3 存储桶的新对象都会自动静态加密。具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）是 Amazon S3 中每个存储桶的默认加密配置。要使用其他类型的加密，您可以指定要在 S3 PUT 请求中使用的服务器端加密类型，也可以在目标存储桶中设置默认加密配置。

如果您想在 PUT 请求中指定不同的加密类型，则可以使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）或具有客户提供的密钥的服务器端加密（SSE-C）。如果您想在目标存储桶中设置不同的默认加密配置，则可以使用 SSE-KMS 或 DSSE-KMS。

您可以使用 S3 控制台、REST API、AWS SDK 和 AWS Command Line Interface（AWS CLI）指定 SSE-S3。有关更多信息，请参阅 为 Amazon S3 存储桶设置默认服务器端加密行为。

使用 S3 控制台

本主题介绍如何使用 AWS Management Console设置或更改对象的加密类型。使用控制台复制对象时，Amazon S3 将按原样复制对象。这意味着，如果对源对象加密，则也会对目标对象加密。可以使用控制台添加或更改对象的加密。

> **Note**
>
> - 如果更改对象的加密，则会创建一个新对象来替换旧对象。如果启用 S3 版本控制，则会创建对象的新版本，而现有对象将变为旧版本。更改属性的角色也会成为新对象（或对象版本）的拥有者。
> - 如果您要更改具有用户定义标签的对象的加密类型，您必须拥有 `s3:GetObjectTagging` 权限。如果您要更改没有用户定义标签但大小超过 16 MB 的对象的加密类型，您还必须拥有 `s3:GetObjectTagging` 权限。
>
>   如果目标存储桶策略拒绝 `s3:GetObjectTagging` 操作，则将更新对象的加密类型，但将从对象中移除用户定义的标签，并且您将收到错误。

更改对象的加密

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Bucket（存储桶）列表中，请选择包含对象的存储桶的名称。
4. 在 Objects（对象）列表中，请选择要为其添加或更改加密的对象的名称。

   将显示对象的详细信息页面，其中有几个部分显示您的对象的属性。
5. 选择属性选项卡。
6. 向下滚动到服务器端加密设置部分，然后选择编辑。
7. 在加密设置下，选择使用存储桶默认加密设置或覆盖存储桶默认加密设置。
8. 如果您选择覆盖默认加密的存储桶设置，请配置以下加密设置。
   - 在加密类型下，选择 Amazon S3 托管式密钥（SSE-S3）。SSE-S3 使用最强的数据块密码之一 [即 256 位高级加密标准（AES-256）] 来加密每个对象。有关更多信息，请参阅 使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。
9. 选择 Save changes（保存更改）。

> **Note**
>
> 此操作将加密应用于所有指定的对象。加密文件夹时，请等待保存操作完成，然后再将新对象添加到文件夹。

使用 REST API

创建对象时（即，上传新对象或复制现有对象时），您可以通过向请求添加 `x-amz-server-side-encryption` 标头来指定您是否希望 Amazon S3 使用 Amazon S3 托管式密钥（SSE-S3）加密您的数据。将标头的值设置为 Amazon S3 支持的加密算法 `AES256`。Amazon S3 通过返回响应标头 `x-amz-server-side-encryption` 来确认已使用 SSE-S3 存储对象。

以下 REST 上传 API 操作接受 `x-amz-server-side-encryption` 请求标头。

- PUT Object
- PUT Object – 复制
- POST 对象
- 开始分段上传

使用分段上传 API 操作上传大型对象时，您可以通过为启动分段上传请求添加 `x-amz-server-side-encryption` 标头来指定服务器端加密。复制现有对象时，不论源对象是否已经加密，都不会加密目标对象，除非您显式请求服务器端加密。

使用 SSE-S3 加密存储对象后，以下 REST API 操作的响应标头将返回 `x-amz-server-side-encryption` 标头。

- PUT Object
- PUT Object – 复制
- POST 对象
- 开始分段上传
- 上传分段
- 上传分段 – 复制
- 完成分段上传
- GET Object

- HEAD Object

> **Note**
>
> 如果您的对象使用 SSE-S3，请不要发送 GET 请求和 HEAD 请求的加密请求标头，否则会收到 HTTP 状态代码 400（错误请求）错误。

使用 AWS SDK

使用 AWS SDK 时，您可以请求 Amazon S3 使用具有 Amazon S3 托管式加密密钥的服务器端加密（SSE-S3）。这部分提供了以多种语言使用 AWS SDK 的示例。有关其他 SDK 的信息，请转到示例代码和库。

Java

当您使用 AWS SDK for Java 上传对象时，可以使用 SSE-S3 为对象加密。要请求服务器端加密，请使用 `ObjectMetadata` 的 `PutObjectRequest` 属性设置 `x-amz-server-side-encryption` 请求标头。当您调用 `AmazonS3Client` 的 `putObject()` 方法时，Amazon S3 将加密并保存数据。

当使用分段上传 API 操作上传对象时，还可以请求 SSE-S3 加密：

- 使用高级别分段上传 API 操作时，请在上传对象时使用 `TransferManager` 方法将服务器端加密应用于对象。可以使用将 `ObjectMetadata` 视为参数的任一上传方法。有关更多信息，请参阅 使用分段上传操作上传对象。
- 当使用低级别分段上传 API 操作时，应在启动分段上传时指定服务器端加密。您通过调用 `ObjectMetadata` 方法添加 `InitiateMultipartUploadRequest.setObjectMetadata()` 属性。有关更多信息，请参阅 使用 AWS SDK（低级别 API）。

不能直接更改对象的加密状态 (加密未加密的对象或解密已加密的对象)。要更改对象的加密状态，请为对象创建一个副本，从而为副本指定所需的加密状态，然后删除原始对象。仅当您显式请求服务器端加密时，Amazon S3 才会加密复制的对象。要通过 Java API 请求加密复制的对象，请使用 `ObjectMetadata` 属性在 `CopyObjectRequest` 中指定服务器端加密。

Example 示例

以下示例演示如何使用 AWS SDK for Java 设置服务器端加密。它展示了如何执行以下任务：

- 使用 SSE-S3 上传新对象。
- 通过为对象创建副本来更改对象的加密状态 (本示例中为加密之前未加密的对象)。
- 检查对象的加密状态。

有关服务器端加密的更多信息，请参阅 使用 REST API。有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.internal.SSEResultBase;
import com.amazonaws.services.s3.model.*;

import java.io.ByteArrayInputStream;

public class SpecifyServerSideEncryption {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyNameToEncrypt = "*** Key name for an object to upload and encrypt
 ***";
        String keyNameToCopyAndEncrypt = "*** Key name for an unencrypted object to
 be encrypted by copying ***";
        String copiedObjectKeyName = "*** Key name for the encrypted copy of the
 unencrypted object ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Upload an object and encrypt it with SSE.
            uploadObjectWithSSEEncryption(s3Client, bucketName, keyNameToEncrypt);

            // Upload a new unencrypted object, then change its encryption state
            // to encrypted by making a copy.
```

```
            changeSSEEncryptionStatusByCopying(s3Client,
                    bucketName,
                    keyNameToCopyAndEncrypt,
                    copiedObjectKeyName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void uploadObjectWithSSEEncryption(AmazonS3 s3Client, String
bucketName, String keyName) {
        String objectContent = "Test object encrypted with SSE";
        byte[] objectBytes = objectContent.getBytes();

        // Specify server-side encryption.
        ObjectMetadata objectMetadata = new ObjectMetadata();
        objectMetadata.setContentLength(objectBytes.length);

objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
        PutObjectRequest putRequest = new PutObjectRequest(bucketName,
                keyName,
                new ByteArrayInputStream(objectBytes),
                objectMetadata);

        // Upload the object and check its encryption status.
        PutObjectResult putResult = s3Client.putObject(putRequest);
        System.out.println("Object \"" + keyName + "\" uploaded with SSE.");
        printEncryptionStatus(putResult);
    }

    private static void changeSSEEncryptionStatusByCopying(AmazonS3 s3Client,
            String bucketName,
            String sourceKey,
            String destKey) {
        // Upload a new, unencrypted object.
        PutObjectResult putResult = s3Client.putObject(bucketName, sourceKey,
"Object example to encrypt by copying");
        System.out.println("Unencrypted object \"" + sourceKey + "\" uploaded.");
```

```
        printEncryptionStatus(putResult);

        // Make a copy of the object and use server-side encryption when storing the
        // copy.
        CopyObjectRequest request = new CopyObjectRequest(bucketName,
                sourceKey,
                bucketName,
                destKey);
        ObjectMetadata objectMetadata = new ObjectMetadata();

 objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
        request.setNewObjectMetadata(objectMetadata);

        // Perform the copy operation and display the copy's encryption status.
        CopyObjectResult response = s3Client.copyObject(request);
        System.out.println("Object \"" + destKey + "\" uploaded with SSE.");
        printEncryptionStatus(response);

        // Delete the original, unencrypted object, leaving only the encrypted copy
 in
        // Amazon S3.
        s3Client.deleteObject(bucketName, sourceKey);
        System.out.println("Unencrypted object \"" + sourceKey + "\" deleted.");
    }

    private static void printEncryptionStatus(SSEResultBase response) {
        String encryptionStatus = response.getSSEAlgorithm();
        if (encryptionStatus == null) {
            encryptionStatus = "Not encrypted with SSE";
        }
        System.out.println("Object encryption status is: " + encryptionStatus);
    }
}
```

.NET

在上传对象时，可指示 Amazon S3 加密对象。要更改现有对象的加密状态，请复制该对象并删除源对象。默认情况下，仅当您显式请求目标对象的服务器端加密时，复制操作才会加密目标。要在 CopyObjectRequest 中指定 SSE-S3，请添加以下内容：

```
ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
```

有关如何复制对象的有效示例，请参阅 使用 AWS SDK。

以下示例将上传对象。在请求中，该示例指示 Amazon S3 加密对象。该示例随后检索对象元数据并验证使用的加密方法。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SpecifyServerSideEncryptionTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for object created ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
                };

                var putResponse = await client.PutObjectAsync(putRequest);
```

```
                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };
                GetObjectMetadataResponse response = await
 client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
 response.ServerSideEncryptionMethod;

                Console.WriteLine("Encryption method used: {0}",
 objectEncryption.ToString());
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
 an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

本主题介绍如何使用 AWS SDK for PHP 版本 3 中的类来将 SSE-S3 添加到您上传到 Amazon S3 的对象。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

要将对象上传到 Amazon S3，请使用 Aws\S3\S3Client::putObject() 方法。要将 `x-amz-server-side-encryption` 请求标头添加到您的上传请求，请使用值 `ServerSideEncryption` 指定 `AES256` 参数，如以下代码示例中所示。有关服务器端加密请求的信息，请参阅 使用 REST API。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

```
$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

// $filepath should be an absolute path to a file on disk.
$filepath = '*** Your File Path ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Upload a file with server-side encryption.
$result = $s3->putObject([
    'Bucket'               => $bucket,
    'Key'                  => $keyname,
    'SourceFile'           => $filepath,
    'ServerSideEncryption' => 'AES256',
]);
```

作为响应，Amazon S3 会返回 `x-amz-server-side-encryption` 标头以及已用于加密对象数据的加密算法的值。

在使用分段上传 API 操作上传大型对象时，您可以为正在上传的对象指定 SSE-S3，如下所示：

- 如果您使用低级别分段上传 API 操作，请在调用 Aws\S3\S3Client::createMultipartUpload() 方法时指定服务器端加密。要向请求添加 `x-amz-server-side-encryption` 请求标头，请使用值 `array` 指定 `ServerSideEncryption` 参数的 `AES256` 密钥。有关低级别分段上传 API 操作的更多信息，请参阅使用 AWS SDK（低级别 API）。
- 当使用高级别分段上传 API 操作时，请使用 CreateMultipartUpload API 操作的 `ServerSideEncryption` 参数来指定服务器端加密。有关将 `setOption()` 方法与高级别分段上传 API 操作结合使用的示例，请参阅使用分段上传操作上传对象。

要确定现有对象的加密状态，请通过调用 Aws\S3\S3Client::headObject() 方法检索对象元数据，如下面的 PHP 代码示例所示。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
```

```
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Check which server-side encryption algorithm is used.
$result = $s3->headObject([
    'Bucket' => $bucket,
    'Key'    => $keyname,
]);
echo $result['ServerSideEncryption'];
```

要更改现有对象的加密状态，请使用 Aws\S3\S3Client::copyObject() 方法复制对象并删除源对象。默认情况下，copyObject() 不会加密目标，除非您通过将值 AES256 用于 ServerSideEncryption 参数，显式请求对目标对象进行服务器端加密。以下 PHP 代码示例将复制对象并向复制的对象添加服务器端加密。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';

$targetBucket = '*** Your Target Bucket Name ***';
$targetKeyname = '*** Your Target Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Copy an object and add server-side encryption.
$s3->copyObject([
    'Bucket'               => $targetBucket,
    'Key'                  => $targetKeyname,
    'CopySource'           => "$sourceBucket/$sourceKeyname",
    'ServerSideEncryption' => 'AES256',
]);
```

有关更多信息，请参阅以下主题：

- 面向 Amazon S3 Aws\S3\S3Client 类的 AWS SDK for PHP
- AWS SDK for PHP 文档

Ruby

在使用 AWS SDK for Ruby 上传对象时，您可以指定使用 SSE-S3 对存储的对象进行静态加密。在您读回对象时，它将自动解密。

下面的 AWS SDK for Ruby 版本 3 示例演示了如何指定对上传到 Amazon S3 的文件进行静态加密。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"
```

```
  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

下面的代码示例演示了如何确定现有对象的加密状态。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data
```

```
  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

如果存储在 Amazon S3 中的对象没有使用服务器端加密，则该方法将返回 null。

要更改现有对象的加密状态，请复制该对象并删除源对象。默认情况下，复制方法不会加密目标，除非您明确请求服务器端加密。您可以通过在选项的哈希参数中指定 server_side_encryption 值来请求对目标对象进行加密，如下面的 Ruby 代码示例所示。此代码示例演示如何复制对象和使用 SSE-S3 加密副本。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
```

```
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 AWS CLI

要在使用 AWS CLI 上传对象时指定 SSE-S3，请使用以下示例。

```
aws s3api put-object --bucket amzn-s3-demo-bucket1 --key object-key-name --server-side-
encryption AES256  --body file path
```

有关更多信息，请参阅 AWS CLI 参考中的 put-object。要在使用 AWS CLI 复制对象时指定 SSE-S3，请参阅 copy-object。

使用 AWS CloudFormation

有关使用 AWS CloudFormation 设置加密的示例，请参阅《AWS CloudFormation 用户指南》的 `AWS::S3::Bucket ServerSideEncryptionRule` 主题中的使用默认加密创建存储桶和通过 AWS KMS 服务器端加密使用 S3 存储桶密钥创建存储桶示例。

使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

服务器端加密是指由接收数据的应用程序或服务在目标位置对数据进行加密。

Amazon S3 对于上传的新对象自动启用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

除非您另行指定，否则默认情况下存储桶使用 SSE-S3 来加密对象。但是，您可以选择将存储桶配置为改为使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）。有关更多信息，请参阅 使用 AWS KMS (SSE-KMS) 指定服务器端加密。

AWS KMS 是一项服务，可将安全、高度可用的硬件和软件结合起来，以提供可扩展到云的密钥管理系统。Amazon S3 使用具有 AWS KMS 的服务器端加密（SSE-KMS）来加密您的 S3 对象数据。此外，当为对象请求 SSE-KMS 时，S3 校验和（作为对象元数据的一部分）将以加密形式存储。有关校验和的更多信息，请参阅检查对象完整性。

如果使用 KMS 密钥，您可以通过 AWS Management Console或 AWS KMS API 使用 AWS KMS 来执行以下操作：

- 集中创建、查看、编辑、监控、启用或禁用、轮换以及安排删除 KMS 密钥。
- 定义控制如何使用和谁可以使用 KMS 密钥的策略。
- 审计它们的使用情况以确认它们被正确使用。AWS KMS API 支持审计，但 AWS KMSAWS Management Console 不支持。

AWS KMS 中的安全控制可帮助您满足与加密相关的合规性要求。您可以利用这些 KMS 密钥来保护在 Amazon S3 存储桶中的数据。将 SSE-KMS 加密用于 S3 存储桶时，AWS KMS keys 必须位于该存储桶所在的同一区域中。

使用 AWS KMS keys 无需支付额外费用。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的 AWS KMS key 概念和 AWS KMS 定价。

权限

要将使用 AWS KMS key 加密的对象上传到 Amazon S3，您需要该密钥的 `kms:GenerateDataKey` 权限。要下载使用 AWS KMS key 加密的对象，您需要 `kms:Decrypt` 权限。有关分段上传所需的 AWS KMS 权限的信息，请参阅分段上传 API 和权限。

> **Important**
>
> 仔细查看在您的 KMS 密钥政策中授予的权限。始终将客户自主管理型 KMS 密钥政策权限仅限制为必须访问相关 AWS KMS 密钥操作的 IAM 主体和 AWS 服务。有关更多信息，请参阅AWS KMS 中的密钥策略。

主题

- AWS KMS keys
- Amazon S3 存储桶密钥
- 需要服务器端加密
- 加密上下文
- 发送对 AWS KMS 加密对象的请求
- 使用 AWS KMS (SSE-KMS) 指定服务器端加密
- 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本

AWS KMS keys

当您使用具有 AWS KMS 的服务器端加密（SSE-KMS）时，您可以使用默认的 AWS 托管式密钥，也可以指定您已创建的客户托管式密钥。AWS KMS 支持信封加密。S3 将 AWS KMS 特征用于信封加密来进一步保护您的数据。信封加密是一种加密方法，它使用数据密钥对明文数据进行加密，然后使用 KMS 密钥对该数据密钥进行加密。有关信封加密的更多信息，请参阅 AWS Key Management Service 开发人员指南中的信封加密。

如果未指定客户托管密钥，首次将使用 SSE-KMS 加密的对象添加到存储桶时，Amazon S3 会自动在 AWS 账户 中创建 AWS 托管式密钥。默认情况下，Amazon S3 将此 KMS 密钥用于 SSE-KMS。

> **Note**
>
> 通过 SSE-KMS（使用 AWS 托管式密钥）加密的对象不能跨账户共享。如果您需要跨账户共享 SSE-KMS 数据，则必须使用来自 AWS KMS 的客户自主管理型密钥。

如果要使用客户托管式密钥进行 SSE-KMS 加密，可以在配置 SSE-KMS 之前创建对称加密客户托管式密钥。然后，为存储桶配置 SSE-KMS 时，请指定现有的客户托管密钥。有关对称加密密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的对称加密 KMS 密钥。

创建客户托管密钥可为您提供更大的灵活性和控制力。例如，您可以创建、轮换和禁用客户托管密钥。您还可以定义访问控制和审核用于保护数据的客户托管密钥。有关客户托管式密钥和 AWS 托管式密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。

> **Note**
>
> 与标准 KMS 密钥不同，当您将服务器端加密与存储在外部密钥存储中的客户托管式密钥结合使用时，您有责任确保密钥材料的可用性和耐久性。有关外部密钥存储及其如何更改责任共担模式的更多信息，请参阅《AWS Key Management Service 开发者指南》中的外部密钥存储。

使用 SSE-KMS 加密进行跨账户操作

在对跨账户操作使用加密时，请注意以下事项：

- 如果未在请求时提供 AWS KMS key Amazon 资源名称（ARN）或别名，也未通过存储桶的默认加密配置提供它们时，则使用 AWS 托管式密钥（`aws/s3`）。
- 如果您使用您的 KMS 密钥所在的相同 AWS 账户中的 AWS Identity and Access Management（IAM）主体上传或访问 S3 对象，则可以使用 AWS 托管式密钥（`aws/s3`）。
- 如果您希望授予对 S3 对象的跨账户访问权限，请使用客户自主管理型密钥。您可以配置客户托管式密钥的策略，以便允许从其他账户进行访问。
- 如果您指定客户托管式 KMS 密钥，我们建议您使用完全限定的 KMS 密钥 ARN。如果您改为使用 KMS 密钥别名，AWS KMS 将解析请求者账户中的密钥。这一行为可能导致使用属于请求者而不是存储桶拥有者的 KMS 密钥来加密数据。
- 您必须指定您（请求者）已被授予 `Encrypt` 权限的密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的允许密钥用户使用 KMS 密钥进行加密操作。

有关何时使用客户自主管理型密钥和 AWS KMS 托管密钥的更多信息，请参阅我是应使用 AWS 托管式密钥 还是使用客户自主管理型密钥来加密 Amazon S3 中的对象？

SSE-KMS 加密工作流程

如果您选择使用 AWS 托管式密钥或客户托管式密钥加密数据，则 AWS KMS 和 Amazon S3 执行以下信封加密操作：

1. Amazon S3 请求明文数据密钥以及使用指定 KMS 密钥加密的数据密钥的副本。
2. AWS KMS 生成数据密钥，使用 KMS 密钥为其进行加密，然后将明文数据密钥和加密的数据密钥发送到 Amazon S3。
3. Amazon S3 使用数据密钥加密数据，并在使用后尽快从内存中删除该明文密钥。
4. Amazon S3 将加密的数据密钥作为元数据与加密数据一起存储。

当请求解密数据时，Amazon S3 和 AWS KMS 将执行以下操作：

1. Amazon S3 在 `Decrypt` 请求中向 AWS KMS 发送加密的数据密钥。
2. AWS KMS 使用相同的 KMS 密钥为加密的数据密钥解密，然后将明文数据密钥返回到 Amazon S3。
3. Amazon S3 使用明文数据密钥解密已加密的数据，并尽快从内存中删除该明文数据密钥。

> **Important**
>
> 在 Amazon S3 中使用 AWS KMS key 进行服务器端加密时，您必须选择对称加密 KMS 密钥。Amazon S3 仅支持对称加密 KMS 密钥。有关这些密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的对称加密 KMS 密钥。

审计 SSE-KMS 加密

要识别指定 SSE-KMS 的请求，您可以使用 Amazon S3 Storage Lens 存储统计管理工具指标中的 All SSE-KMS requests（所有 SSE-KMS 请求）和 % all SSE-KMS requests（所有 SSE-KMS 请求的百分比）指标。S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。还可以使用启用 SSE-KMS 的存储桶计数和启用 SSE-KMS 的存储桶百分比，来了解使用 SSE-KMS 进行默认存储桶加密的存储桶计数。有关更多信息，请参阅使用 S3 Storage Lens 存储统计管理工具访问存储活动和使用情况。有关指标的完整列表，请参阅 S3 Storage Lens 存储统计管理工具指标词汇表。

要审计为 SSE-KMS 加密数据使用 AWS KMS 密钥的情况，您可以使用 AWS CloudTrail 日志。您可以深入了解自己的加密操作，例如 GenerateDataKey 和 Decrypt。CloudTrail 支持多种属性值来筛选您的搜索，包括事件名称、用户名和事件源。

Amazon S3 存储桶密钥

当您配置使用 AWS KMS 的服务器端加密（SSE-KMS）时，您可以将存储桶配置为使用 S3 存储桶密钥实现 SSE-KMS。使用 SSE-KMS 的存储桶级别密钥可以通过减少从 Amazon S3 到 AWS KMS 的流量请求，从而使您可以将 AWS KMS 请求成本最高降低 99%。

当您将存储桶配置为使用 S3 存储桶密钥对新对象实现 SSE-KMS 时，AWS KMS 会生成存储桶级别密钥，该密钥用于为存储桶中的对象创建唯一的数据密钥。此 S3 存储桶密钥在 Amazon S3 内限时使用，从而进一步减少了 Amazon S3 向 AWS KMS 发出请求以完成加密操作的需求。有关使用 S3 存储桶密钥的更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

需要服务器端加密

如果要求对特定 Amazon S3 存储桶中的所有对象进行服务器端加密，请使用存储桶策略。例如，如果请求不包含用于请求服务器端加密（SSE-KMS）的 `x-amz-server-side-encryption-aws-kms-key-id` 标头，则下面的存储桶策略将拒绝所有人的上传对象（`s3:PutObject`）权限。

```
{
   "Version":"2012-10-17",
   "Id":"PutObjectPolicy",
   "Statement":[{
         "Sid":"DenyObjectsThatAreNotSSEKMS",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*",
         "Condition":{
            "Null":{
               "s3:x-amz-server-side-encryption-aws-kms-key-id":"true"
            }
         }
      }
   ]
}
```

如要求使用特定 AWS KMS key 对存储桶中的对象进行加密，可以使用 `s3:x-amz-server-side-encryption-aws-kms-key-id` 条件键。要指定 KMS 密钥，必须使用

`arn:aws:kms:region:acct-id:key/key-id` 格式的键 Amazon 资源名称（ARN）。AWS Identity and Access Management 不验证 `s3:x-amz-server-side-encryption-aws-kms-key-id` 的字符串是否存在。

> **Note**
>
> 在上传对象时，可以使用 `x-amz-server-side-encryption-aws-kms-key-id` 标头指定 KMS 密钥，也可以依赖您的默认存储桶加密配置。如果 PutObject 请求在 `x-amz-server-side-encryption` 标头中指定 `aws:kms`，但未指定 `x-amz-server-side-encryption-aws-kms-key-id` 标头，则 Amazon S3 会假定您要使用 AWS 托管式密钥。无论如何，Amazon S3 用于对象加密的 AWS KMS 密钥 ID 必须与策略中的 AWS KMS 密钥 ID 匹配，否则 Amazon S3 会拒绝请求。

要查看 Amazon S3 特定条件键的列表，请参阅《Service Authorization Reference》中的 Condition keys for Amazon S3。

加密上下文

加密上下文是一组键值对，其中包含有关数据的其他上下文信息。加密上下文没有加密。在为加密操作指定加密上下文时，Amazon S3 必须指定与解密操作相同的加密上下文。否则，解密将失败。AWS KMS 会将加密上下文用作其他已经过验证的数据 (AAD) 以支持经过身份验证的加密。有关加密上下文的更多信息，请参阅 AWS Key Management Service 开发人员指南中的加密上下文。

默认情况下，Amazon S3 使用对象或存储桶 Amazon 资源名称（ARN）作为加密上下文对：

- 如果您在不启用 S3 存储桶密钥的情况下使用 SSE-KMS，则将对象 ARN 用作加密上下文。

```
arn:aws:s3:::object_ARN
```

- 如果您使用 SSE-KMS 并启用 S3 存储桶密钥，则将存储桶 ARN 用作加密上下文。有关 S3 存储桶的更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

```
arn:aws:s3:::bucket_ARN
```

您可以选择使用 s3:PutObject 请求中的 `x-amz-server-side-encryption-context` 标头提供其他的加密上下文对。但是，由于加密上下文未加密，请确保它不包含敏感信息。Amazon S3 将此额外的密钥对与默认加密上下文一起存储。当处理 PUT 请求时，Amazon S3 会将 `aws:s3:arn` 的默认加密上下文附加到您提供的上下文中。

您可以使用加密上下文来标识和分类加密操作。您还可以使用默认的加密上下文 ARN 值来跟踪 AWS CloudTrail 中的相关请求，方法是通过查看哪个 Amazon S3 ARN 与哪个加密密钥一起使用。

在 CloudTrail 日志文件的 `requestParameters` 字段中，加密上下文与以下内容类似。

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket1/file_name"
}
```

在将 SSE-KMS 与可选的 S3 存储桶密钥特征一起使用时，加密上下文值是存储桶的 ARN。

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket1"
}
```

发送对 AWS KMS 加密对象的请求

> **Important**
>
> 对 AWS KMS 加密对象的所有 GET 和 PUT 请求必须使用安全套接字层协议（SSL）和传输层安全性协议（TLS）发出。还必须使用有效的凭证对请求进行签名，例如 AWS 签名版本 4（或 AWS 签名版本 2）。

AWS 签名版本 4 是将身份验证信息添加到通过 HTTP 发送的 AWS 请求的过程。出于安全考虑，大多数 AWS 请求都必须使用访问密钥（包括访问密钥 ID 和秘密访问密钥）进行签名。这两个密钥通常称为您的安全凭证。有关更多信息，请参阅对请求进行身份验证（AWS Signature Version 4）和 Signature Version 4 签名过程。

> **Important**
>
> 如果对象使用 SSE-KMS，则不应对 GET 请求和 HEAD 请求发送加密请求标头。否则，您会收到 HTTP 400 错误请求错误。

主题

- 使用 AWS KMS (SSE-KMS) 指定服务器端加密
- 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本

使用 AWS KMS (SSE-KMS) 指定服务器端加密

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

默认情况下，所有 Amazon S3 存储桶都配置了加密，所有上传到 S3 存储桶的新对象都会自动静态加密。具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）是 Amazon S3 中每个存储桶的默认加密配置。要使用其他类型的加密，您可以指定要在 S3 PUT 请求中使用的服务器端加密类型，也可以在目标存储桶中设置默认加密配置。

如果您想在 PUT 请求中指定不同的加密类型，则可以使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）或具有客户提供的密钥的服务器端加密（SSE-C）。如果您想在目标存储桶中设置不同的默认加密配置，则可以使用 SSE-KMS 或 DSSE-KMS。

当您上传新对象或复制现有对象时，您可以应用加密。

您可以使用 Amazon S3 控制台、REST API 操作、AWS SDK 和 AWS Command Line Interface（AWS CLI）指定 SSE-KMS。有关更多信息，请参阅以下主题。

> **Note**
>
> 您可以在 Amazon S3 中使用多区域 AWS KMS keys。但是，Amazon S3 目前将多区域密钥视为单区域密钥，且不使用密钥的多区域特征。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的使用多区域密钥。

> **Note**
>
> 如果您希望使用其它账户拥有的 KMS 密钥，您必须有权使用该密钥。有关 KMS 密钥的跨账户权限的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建其他账户可以使用的 KMS 密钥。

使用 S3 控制台

本主题描述如何使用 Amazon S3 控制台，将对象的加密类型设置或更改为使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）。

> **Note**
>
> - 如果更改对象的加密，则会创建一个新对象来替换旧对象。如果启用 S3 版本控制，则会创建对象的新版本，而现有对象将变为旧版本。更改属性的角色也会成为新对象（或对象版本）的拥有者。
> - 如果您要更改具有用户定义标签的对象的加密类型，您必须拥有 `s3:GetObjectTagging` 权限。如果您要更改没有用户定义标签但大小超过 16 MB 的对象的加密类型，您还必须拥有 `s3:GetObjectTagging` 权限。
>
>   如果目标存储桶策略拒绝 `s3:GetObjectTagging` 操作，则将更新对象的加密类型，但将从对象中移除用户定义的标签，并且您将收到错误。

添加或更改对象的加密

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Bucket（存储桶）列表中，请选择包含对象的存储桶的名称。
4. 在 Objects（对象）列表中，请选择要为其添加或更改加密的对象的名称。

   将显示对象的详细信息页面，其中有几个部分显示您的对象的属性。
5. 选择属性选项卡。
6. 向下滚动到服务器端加密设置部分，然后选择编辑。

Edit server-side encryption (编辑服务器端加密) 页面随即打开。

7. 在服务器端加密下，对于加密设置，选择覆盖默认的加密存储桶设置。
8. 在加密类型下，选择具有 AWS Key Management Service 密钥的服务器端加密（SSE-KMS）。

> **Important**
>
> 如果您将 SSE-KMS 选项用于默认加密配置，则您将受到 AWS KMS 的每秒请求数（RPS）限额限制。有关 AWS KMS 限额以及如何请求增加限额的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的限额。

9. 在 AWS KMS 密钥下，执行以下操作以选择您的 KMS 密钥：

   - 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 中进行选择，然后从可用密钥的列表中选择您的 KMS 密钥。

     AWS 托管式密钥（aws/s3）和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。
   - 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
   - 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

     有关创建 AWS KMS key 的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

> **Important**
>
> 您只能使用与存储桶所在相同的 AWS 区域中可用的 KMS 密钥。Amazon S3 控制台仅列出与存储桶位于同一区域中的前 100 个 KMS 密钥。要使用未列出的 KMS 密钥，您必须输入 KMS 密钥 ARN。如果您希望使用其他账户拥有的 KMS 密钥，则必须首先有权使用该密钥，然后必须输入相应的 KMS 密钥 ARN。
> Amazon S3 仅支持对称加密 KMS 密钥，不支持非对称 KMS 密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的确定对称和非对称 KMS 密钥。

10. 选择 Save Changes（保存更改）。

> **Note**
>
> 此操作将加密应用于所有指定的对象。加密文件夹时，请等待保存操作完成，然后再将新对象添加到文件夹。

使用 REST API

创建对象时（即上传新对象或复制现有对象时），您可以指定使用具有 AWS KMS keys 的服务器端加密（SSE-KMS）来加密数据。为此，请将 `x-amz-server-side-encryption` 标头添加到请求。将标头的值设置为加密算法 `aws:kms`。Amazon S3 通过返回响应标头 `x-amz-server-side-encryption` 来确认已使用 SSE-KMS 存储您的对象。

如果您指定值为 `aws:kms` 的 `x-amz-server-side-encryption` 标头，则还可以使用以下请求标头：

- `x-amz-server-side-encryption-aws-kms-key-id`
- `x-amz-server-side-encryption-context`
- `x-amz-server-side-encryption-bucket-key-enabled`

主题

- 支持 SSE-KMS 的 Amazon S3 REST API 操作
- 加密上下文（x-amz-server-side-encryption-context）
- AWS KMS 密钥 ID（x-amz-server-side-encryption-aws-kms-key-id）
- S3 桶密钥（x-amz-server-side-encryption-aws-bucket-key-enabled）

支持 SSE-KMS 的 Amazon S3 REST API 操作

以下 REST API 操作接受 `x-amz-server-side-encryption`、`x-amz-server-side-encryption-aws-kms-key-id` 和 `x-amz-server-side-encryption-context` 请求标头。

- PutObject – 使用 PUT API 操作上传数据时，您可以指定这些请求标头。
- CopyObject – 复制对象时，您同时具有源对象和目标对象。如果使用 `CopyObject` 操作传递 SSE-KMS 标头，这些标头仅应用于目标对象。复制现有对象时，不论源对象是否已加密，都不会对目标对象加密，除非您显式请求服务器端加密。

- POST Object – 使用 POST 操作上传对象时，可在表单字段（而不是在请求标头）中提供相同的信息。
- CreateMultipartUpload – 使用分段上传 API 操作上传大型对象时，可以指定这些标头。您可以在 CreateMultipartUpload 请求中指定这些标头。

使用服务器端加密存储对象时，以下 REST API 操作的响应标头将返回 x-amz-server-side-encryption 标头。

- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - 如果您不让使用安全套接字协议（SSL）、传输层安全性协议（TLS）或签名版本 4 针对受 AWS KMS 保护的对象发出所有 GET 和 PUT 请求，则这些请求都将失败。
> - 如果对象使用 SSE-KMS，则不应对 GET 请求和 HEAD 请求发送加密请求标头，否则将显示 HTTP 400 BadRequest 错误。

加密上下文（**x-amz-server-side-encryption-context**）

如果您指定 x-amz-server-side-encryption:aws:kms，Amazon S3 API 将支持带有 x-amz-server-side-encryption-context 标头的加密上下文。加密上下文是一组键值对，其中包含有关数据的其他上下文信息。

Amazon S3 会自动使用对象或存储桶 Amazon Resource Name（ARN）作为加密上下文对。如果您在不启用 S3 存储桶密钥的情况下使用 SSE-KMS，则将对象 ARN 用作加密上下文；例如

`arn:aws:s3:::object_ARN`。但是，如果您使用 SSE-KMS 并启用 S3 存储桶密钥，则将存储桶 ARN 用于加密上下文；例如 `arn:aws:s3:::bucket_ARN`。

您可以选择使用 `x-amz-server-side-encryption-context` 标头提供其他的加密上下文对。但是，由于加密上下文未加密，请确保其中不包含敏感信息。Amazon S3 将此额外的密钥对与默认加密上下文一起存储。

有关 Amazon S3 中加密上下文的信息，请参阅 加密上下文。有关加密上下文的一般信息，请参阅 AWS Key Management Service 开发人员指南中的 AWS Key Management Service 概念 - 加密上下文。

AWS KMS 密钥 ID（**`x-amz-server-side-encryption-aws-kms-key-id`**）

您可以使用 `x-amz-server-side-encryption-aws-kms-key-id` 标头指定用于保护数据的客户自主管理型密钥的 ID。如果您指定 `x-amz-server-side-encryption:aws:kms` 标头但未提供 `x-amz-server-side-encryption-aws-kms-key-id` 标头，Amazon S3 将使用 AWS 托管式密钥（`aws/s3`）来保护数据。如果要使用客户托管密钥，则必须提供客户托管密钥的 `x-amz-server-side-encryption-aws-kms-key-id` 标头。

> **Important**
>
> 在 Amazon S3 中使用 AWS KMS key 进行服务器端加密时，您必须选择对称加密 KMS 密钥。Amazon S3 仅支持对称加密 KMS 密钥。有关这些密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的对称加密 KMS 密钥。

S3 桶密钥（**`x-amz-server-side-encryption-aws-bucket-key-enabled`**）

您可以使用 `x-amz-server-side-encryption-aws-bucket-key-enabled` 请求标头在对象级别启用或禁用 S3 存储桶密钥。S3 存储桶密钥通过减少从 Amazon S3 到 AWS KMS 的请求流量来降低您的 AWS KMS 请求成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

如果您指定 `x-amz-server-side-encryption:aws:kms` 标头但未提供 `x-amz-server-side-encryption-aws-bucket-key-enabled` 标头，则您的对象将使用目标存储桶的 S3 存储桶密钥设置来加密对象。有关更多信息，请参阅 在对象级别配置 S3 存储桶密钥 。

使用 AWS CLI

要使用以下示例 AWS CLI 命令，请将 *`user input placeholders`* 替换为您自己的信息。

当您上传新对象或复制现有对象时，可以指定使用具有 AWS KMS 密钥的服务器端加密来加密数据。为此，请将 `--server-side-encryption aws:kms` 标头添加到请求。使用 `--ssekms-key-id example-key-id` 添加您创建的客户托管式 AWS KMS 密钥。如果您指定 `--server-side-encryption aws:kms`，但未提供 AWS KMS 密钥 ID，Amazon S3 将使用 AWS 托管式密钥。

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --
server-side-encryption aws:kms --ssekms-key-id example-key-id --ssekms-encryption-
context example-encryption-context --body filepath
```

您还可以通过添加 `--bucket-key-enabled` 或 `--no-bucket-key-enabled`，在 `put-object` 或 `copy-object` 操作中启用或禁用 S3 存储桶密钥。S3 桶密钥可以通过减少从 Amazon S3 到 AWS KMS 的请求流量来降低您的 AWS KMS 请求成本。有关更多信息，请参阅使用 S3 存储桶密钥降低 SSE-KMS 的成本。

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --server-
side-encryption aws:kms --bucket-key-enabled --body filepath
```

您可以将对象从源存储桶复制到新存储桶，并指定 SSE-KMS 加密。

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket/example-object-key --
bucket amzn-s3-demo-bucket2 --key example-object-key --server-side-encryption aws:kms
 --sse-kms-key-id example-key-id --ssekms-encryption-context example-encryption-context
```

使用 AWS SDK

使用 AWS SDK 时，您可以请求 Amazon S3 使用 AWS KMS keys 进行服务器端加密。以下示例展示了如何将 SSE-KMS 与适用于 Java 和 .NET 的 AWS SDK 结合使用。有关其它 SDK 的信息，请参阅 AWS 开发人员中心上的示例代码和库。

> **Important**
>
> 在 Amazon S3 中使用 AWS KMS key 进行服务器端加密时，您必须选择对称加密 KMS 密钥。Amazon S3 仅支持对称加密 KMS 密钥。有关这些密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的对称加密 KMS 密钥。

`CopyObject` 操作

在复制对象时，您添加相同的请求属性（`ServerSideEncryptionMethod` 和 `ServerSideEncryptionKeyManagementServiceKeyId`）来请求 Amazon S3 使用 AWS KMS key。有关复制对象的更多信息，请参阅 复制、移动和重命名对象。

PUT 操作

Java

当使用AWS SDK for Java 上传对象时，您可以通过添加 `SSEAwsKeyManagementParams` 属性来请求 Amazon S3 使用 AWS KMS key，如以下请求所示：

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams());
```

在这种情况下，Amazon S3 使用 AWS 托管式密钥（`aws/s3`）。有关更多信息，请参阅 使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。您可以选择创建对称加密 KMS 密钥，并在请求中指定该密钥，如以下示例所示：

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new
 SSEAwsKeyManagementParams(keyID));
```

有关创建客户托管密钥的更多信息，请参阅 AWS Key Management Service 开发人员指南中的对 AWS KMS API 进行编程。

有关上传对象的工作代码示例，请参阅以下主题。要使用这些示例，您必须更新这些代码示例并提供加密信息，如上述代码片段所示。

- 有关在单个操作中上传对象，请参阅 上传对象。
- 有关使用高级或低级分段上传 API 操作的分段上传，请参阅使用分段上传操作上传对象。

.NET

当使用AWS SDK for .NET 上传对象时，您可以通过添加 `ServerSideEncryptionMethod` 属性来请求 Amazon S3 使用 AWS KMS key，如以下请求所示：

```
PutObjectRequest putRequest = new PutObjectRequest
```

```
 {
     BucketName = amzn-s3-demo-bucket,
     Key = keyName,
     // other properties
     ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS
 };
```

在这种情况下，Amazon S3 使用 AWS 托管式密钥。有关更多信息，请参阅 使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。您可以选择创建自己的对称加密客户自主管理型密钥，并在请求中指定该密钥，如以下示例所示：

```
PutObjectRequest putRequest1 = new PutObjectRequest
{
  BucketName = amzn-s3-demo-bucket,
  Key = keyName,
  // other properties
  ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS,
  ServerSideEncryptionKeyManagementServiceKeyId = keyId
};
```

有关创建客户托管密钥的更多信息，请参阅 AWS Key Management Service 开发人员指南中的对 AWS KMS API 进行编程。

有关上传对象的工作代码示例，请参阅以下主题。要使用这些示例，您必须更新这些代码示例并提供加密信息，如上述代码片段所示。

- 有关在单个操作中上传对象，请参阅 上传对象。
- 有关使用高级或低级分段上传 API 操作的分段上传，请参阅使用分段上传操作上传对象。

预签名 URL

Java

在为使用 AWS KMS key加密的对象创建预签名 URL 时，您必须显式指定签名版本 4，如以下示例所示：

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setSignerOverride("AWSS3V4SignerType");
AmazonS3Client s3client = new AmazonS3Client(
        new ProfileCredentialsProvider(), clientConfiguration);
```

```
...
```

有关代码示例，请参阅 使用预签名 URL 共享对象。

.NET

在为使用 AWS KMS key加密的对象创建预签名 URL 时，您必须显式指定签名版本 4，如以下示例所示：

```
AWSConfigs.S3Config.UseSignatureVersion4 = true;
```

有关代码示例，请参阅 使用预签名 URL 共享对象。

使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本

Amazon S3 存储桶密钥降低了具有 AWS Key Management Service（AWS KMS）密钥的 Amazon S3 服务器端加密（SSE-KMS）的成本。使用 SSE-KMS 的存储桶级别密钥可以通过减少从 Amazon S3 到 AWS KMS 的请求流量，从而使您可以将 AWS KMS 请求成本最高降低 99%。只需在 AWS Management Console中单击几下，无需对客户端应用程序进行任何更改，您就可以将存储桶配置为使用 S3 存储桶密钥对新对象进行 SSE-KMS 加密。

> **Note**
>
> 使用 AWS Key Management Service（AWS KMS）密钥的双层服务器端加密（DSSE-KMS）不支持 S3 存储桶密钥。

SSE-KMS 的 S3 存储桶密钥

访问使用 SSE-KMS 加密的数百万或数十亿个对象的工作负载可以生成大量到 AWS KMS 的请求。当您在没有 S3 存储桶密钥的情况下使用 SSE-KMS 保护数据时，Amazon S3 会为每个对象使用单独的 AWS KMS 数据密钥。在这种情况下，每次对 KMS 加密的对象发出请求时，Amazon S3 都会调用 AWS KMS。有关 SSE-KMS 工作原理的信息，请参阅 使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。

当您将存储桶配置为使用 S3 存储桶密钥进行 SSE-KMS 加密时，AWS 会从 AWS KMS 生成生存期较短的存储桶级密钥，然后暂时将其保留在 S3 中。此存储桶级密钥将在新对象的生命周期中为其创建数据密钥。S3 存储桶密钥在 Amazon S3 内限时使用，从而减少了 S3 向 AWS KMS 发出请求以完成加

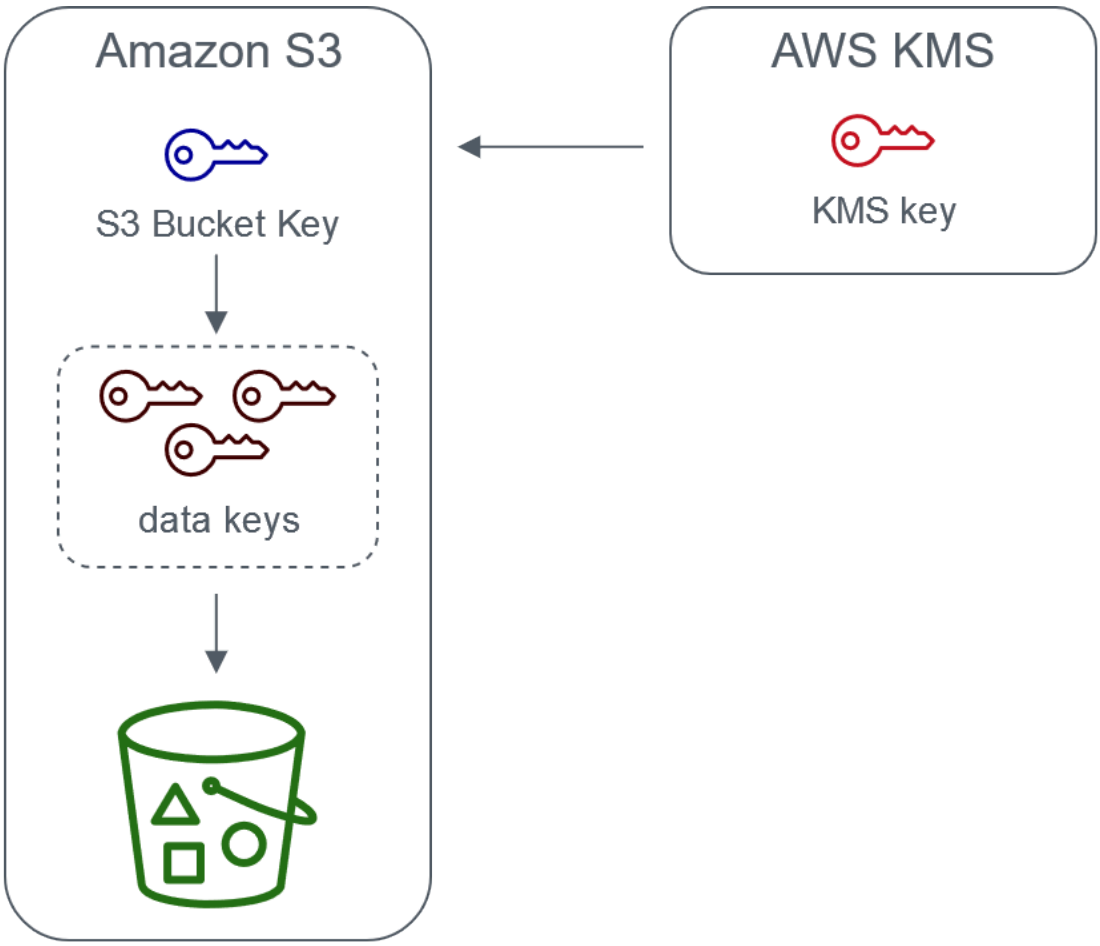
密操作的需求。这样可以减少从 S3 到 AWS KMS 的流量，使您能够在 Amazon S3 中访问 AWS KMS 加密的对象，所需成本仅为以前的一小部分。

每个请求者至少获取一次唯一存储桶级密钥，以确保在 AWS KMS CloudTrail 事件中捕获请求者对密钥的访问权限。当调用方使用不同的角色或账户，或使用具有不同范围限定策略的相同角色时，Amazon S3 会将其视为不同的请求者。AWS KMS 节省的请求反映了请求者的数量、请求模式和所请求对象的相对年限。例如，减少请求者数量，在有限的时间窗口内请求多个对象，并使用相同的存储桶级密钥进行加密，可以节省更多费用。

> **Note**
>
> 利用 S3 存储桶密钥，可通过使用桶级密钥减少向 AWS KMS 发出的面向 `Encrypt`、`GenerateDataKey` 和 `Decrypt` 操作的请求数，从而节省 AWS KMS 请求成本。根据设计，利用此存储桶级密钥的后续请求不会产生 AWS KMS API 请求或根据 AWS KMS 密钥策略验证访问权限。

配置 S3 存储桶密钥时，存储桶中已存在的对象不使用 S3 存储桶密钥。要为现有对象配置 S3 存储桶密钥，可以使用 `CopyObject` 操作。有关更多信息，请参阅 在对象级别配置 S3 存储桶密钥 。

Amazon S3 将仅为由同一 AWS KMS key 加密的对象共享 S3 存储桶密钥。S3 存储桶密钥与 AWS KMS 创建的 KMS 密钥、导入的密钥材料以及由自定义密钥存储库支持的密钥材料兼容。



Server-side encryption with AWS Key Management service using an S3 Bucket Key

配置 S3 存储桶密钥

您可以通过 Amazon S3 控制台、AWS SDK、AWS CLI 或 REST API 将存储桶配置为使用 S3 存储桶密钥对新对象进行 SSE-KMS 加密。在您的存储桶上启用 S3 存储桶密钥后，使用其他指定 SSE-KMS 密钥上传的对象将使用其自己的 S3 存储桶密钥。无论您的 S3 存储桶密钥设置如何，您都可以在请求中包含带 `true` 或 `false` 值的 `x-amz-server-side-encryption-bucket-key-enabled` 标头，以覆盖存储桶设置。

在将存储桶配置为使用 S3 存储桶密钥之前，请查看 启用 S3 存储桶密钥之前需要注意的更改。

使用 Amazon S3 控制台配置 S3 存储桶密钥

创建新存储桶时，您可以将存储桶配置为使用 S3 存储桶密钥对新对象进行 SSE-KMS 加密。您还可以通过更新存储桶属性，从而将现有存储桶配置为使用 S3 存储桶密钥对新对象进行 SSE-KMS 加密。

有关更多信息，请参阅 将存储桶配置为将 S3 存储桶密钥与 SSE-KMS 结合使用于新对象。

REST API、AWS CLI 和 AWS SDK 支持 S3 存储桶密钥

您可以使用 REST API、AWS CLI 或 AWS SDK 将存储桶配置为使用 S3 存储桶密钥对新对象进行 SSE-KMS 加密。您还可以在对象级别启用 S3 存储桶密钥。

有关更多信息，请参阅下列内容：

- 在对象级别配置 S3 存储桶密钥
- 将存储桶配置为将 S3 存储桶密钥与 SSE-KMS 结合使用于新对象

以下 API 操作对于 SSE-KMS 支持 S3 存储桶密钥：

- PutBucketEncryption
  - `ServerSideEncryptionRule` 接受用于启用和禁用 S3 存储桶密钥的 `BucketKeyEnabled` 参数。
- GetBucketEncryption
  - `ServerSideEncryptionRule` 返回 `BucketKeyEnabled` 的设置。
- PutObject、CopyObject、CreateMultipartUpload 和 POST 对象
  - `x-amz-server-side-encryption-bucket-key-enabled` 请求标头在对象级别启用或禁用 S3 存储桶密钥。
- HeadObject、GetObject、UploadPartCopy、UploadPart 和 CompleteMultipartUpload
  - `x-amz-server-side-encryption-bucket-key-enabled` 响应标头指示是否为对象启用或禁用了 S3 存储桶密钥。

使用 AWS CloudFormation

在 AWS CloudFormation 中，`AWS::S3::Bucket` 资源包括名为 `BucketKeyEnabled` 的加密属性，您可以使用该属性来启用或禁用 S3 存储桶密钥。

有关更多信息，请参阅 使用 AWS CloudFormation。

启用 S3 存储桶密钥之前需要注意的更改

在启用 S3 存储桶密钥之前，请注意以下相关更改：

IAM 或 AWS KMS 密钥策略

如果您现有的 AWS Identity and Access Management（IAM）策略或 AWS KMS 密钥策略使用您的对象 Amazon 资源名称（ARN）作为加密上下文来优化或限制对 KMS 密钥的访问，则这些策略将不使

用 S3 存储桶密钥。S3 存储桶密钥使用存储桶 ARN 作为加密上下文。在启用 S3 存储桶密钥之前，请更新 IAM 策略或 AWS KMS 密钥策略，以将存储桶 ARN 用作加密上下文。

有关加密上下文和 S3 存储桶密钥的更多信息，请参阅加密上下文。

AWS KMS 的 CloudTrail 事件

启用 S3 存储桶密钥后，AWS KMS CloudTrail 事件会记录存储桶 ARN 而不是对象 ARN。此外，您在日志中看到的 SSE-KMS 对象的 KMS CloudTrail 事件较少。因为 Amazon S3 中的密钥材料是有时间限制的，所以对 AWS KMS 的请求减少。

将 S3 存储桶密钥与复制功能结合使用

您可以将 S3 存储桶密钥与同区域复制（SRR）和跨区域复制（CRR）结合使用。

当 Amazon S3 复制加密对象时，它通常会在目标存储桶中保留副本对象的加密设置。但是，如果源对象未加密且目标存储桶使用默认加密或 S3 存储桶密钥，则 Amazon S3 会使用目标存储桶的配置加密对象。

以下示例说明了 S3 存储桶密钥如何与复制结合使用。有关更多信息，请参阅 复制加密对象（SSE-C、SSE-S3、SSE-KMS、DSSE-KMS）。

Example 示例 1 – 源对象使用 S3 存储桶密钥；目标存储桶使用默认加密

如果源对象使用 S3 存储桶密钥，但目标存储桶将默认加密与 SSE-KMS 结合使用，则副本对象将在目标存储桶中维护其 S3 存储桶密钥加密设置。目标存储桶仍将默认加密与 SSE-KMS 结合使用。

Example 示例 2 – 源对象未加密；目标存储桶将 S3 存储桶密钥与 SSE-KMS 结合使用

如果源对象未加密，而目标存储桶将 S3 存储桶密钥与 SSE-KMS 结合使用，则将通过在目标存储桶中将 S3 存储桶密钥与 SSE-KMS 结合使用来加密复制的对象。这将导致源对象的 ETag 与副本对象的 ETag 不同。您必须更新使用 ETag 的应用程序以应对这种差异。

使用 S3 存储桶密钥

有关启用和使用 S3 存储桶密钥的更多信息，请参阅以下各部分：

- 将存储桶配置为将 S3 存储桶密钥与 SSE-KMS 结合使用于新对象
- 在对象级别配置 S3 存储桶密钥

- 查看 S3 存储桶密钥的设置

将存储桶配置为将 S3 存储桶密钥与 SSE-KMS 结合使用于新对象

当您配置具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）时，您可以将存储桶配置为使用 S3 存储桶密钥对新对象进行 SSE-KMS 加密。S3 存储桶密钥可减少从 Amazon S3 到 AWS KMS 的请求流量，从而降低 SSE-KMS 的成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

您可以使用 Amazon S3 控制台、REST API、AWS SDK、AWS Command Line Interface（AWS CLI）或 AWS CloudFormation 将存储桶配置为使用 S3 存储桶密钥对新对象进行 SSE-KMS 加密。如果要为现有对象启用或禁用 S3 存储桶密钥，则可以使用 `CopyObject` 操作。有关更多信息，请参阅在对象级别配置 S3 存储桶密钥 和使用 S3 分批操作加密具有 S3 Bucket 密钥的对象。

当为源存储桶或目标存储桶启用 S3 存储桶密钥时，加密上下文将是存储桶 Amazon Resource Name（ARN），而不是对象 ARN，例如 `arn:aws:s3:::`*`bucket_ARN`*。您需要更新 IAM 策略才能将存储桶 ARN 用于加密上下文。有关更多信息，请参阅 S3 存储桶密钥和复制。

以下示例说明了 S3 存储桶密钥如何与复制结合使用。有关更多信息，请参阅 复制加密对象（SSE-C、SSE-S3、SSE-KMS、DSSE-KMS）。

先决条件

在将存储桶配置为使用 S3 存储桶密钥之前，请查看 启用 S3 存储桶密钥之前需要注意的更改。

使用 S3 控制台

在 S3 控制台中，您可以为新存储桶或现有存储桶启用或禁用 S3 存储桶密钥。S3 控制台中的对象从存储桶配置中继承其 S3 存储桶密钥设置。当您为存储桶启用 S3 存储桶密钥时，您上传到存储桶的新对象将利用 S3 存储桶密钥进行 SSE-KMS 加密。

在启用了 S3 存储桶密钥的存储桶中上传，复制或修改对象

如果您在启用了 S3 存储桶密钥的存储桶中上传、修改或复制对象，则该对象的 S3 存储桶密钥设置可能会更新以与存储桶配置保持一致。

如果对象已启用 S3 存储桶密钥，则在复制或修改对象时，该对象的 S3 存储桶密钥设置不会更改。但是，如果您修改或复制未启用 S3 存储桶密钥的对象，并且目标存储桶具有 S3 存储桶密钥配置，则该对象将继承目标存储桶的 S3 存储桶密钥设置。例如，如果源对象尚未启用 S3 存储桶密钥，但目标存储桶已启用 S3 存储桶密钥，则为该对象启用 S3 存储桶密钥。

在创建新存储桶时启用 S3 存储桶密钥

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 请选择 Create bucket（创建存储桶）。
4. 输入存储桶名称，然后选择您的 AWS 区域。
5. 在默认加密下，对于加密密钥类型，选择 AWS Key Management Service 密钥（SSE-KMS）。
6. 在 AWS KMS 密钥下，执行以下操作以选择您的 KMS 密钥：

   - 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 中进行选择，然后从可用密钥的列表中选择您的 KMS 密钥。

     AWS 托管式密钥（aws/s3）和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。
   - 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
   - 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

     有关创建 AWS KMS key 的更多信息，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。
7. 在 Bucket Key（存储桶密钥）下，请选择 Enable（启用）。
8. 请选择 Create bucket（创建存储桶）。

   Amazon S3 创建启用了 S3 存储桶密钥的存储桶。您上传到存储桶的新对象将使用 S3 存储桶密钥。

   要禁用 S3 存储桶密钥，请按照前面的步骤操作，然后选择 Disable（禁用）。

为现有存储桶启用 S3 存储桶密钥

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（存储桶）列表中，请选择要为其启用 S3 存储桶密钥的存储桶。
4. 选择属性选项卡。

5. 在默认加密下，选择编辑。
6. 在默认加密下，对于加密密钥类型，选择 AWS Key Management Service 密钥（SSE-KMS）。
7. 在 AWS KMS 密钥下，执行以下操作以选择您的 KMS 密钥：

   - 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 中进行选择，然后从可用密钥的列表中选择您的 KMS 密钥。

     AWS 托管式密钥（aws/s3）和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。
   - 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
   - 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

     有关创建 AWS KMS key 的更多信息，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。
8. 在 Bucket Key（存储桶密钥）下，请选择 Enable（启用）。
9. 选择 Save Changes（保存更改）。

   Amazon S3 为添加到存储桶中的新对象启用 S3 存储桶密钥。现有对象未使用 S3 存储桶密钥。要为现有对象配置 S3 存储桶密钥，可以使用 CopyObject 操作。有关更多信息，请参阅 在对象级别配置 S3 存储桶密钥 。

   要禁用 S3 存储桶密钥，请按照前面的步骤操作，然后选择 Disable（禁用）。

使用 REST API

您可以使用 PutBucketEncryption 为存储桶启用或禁用 S3 存储桶密钥。要使用 PutBucketEncryption 配置 S3 存储桶密钥，请使用 ServerSideEncryptionRule 数据类型，其中包括使用 SSE-KMS 进行的默认加密。您还可以通过客户托管密钥的 KMS 密钥 ID 来选择使用客户托管密钥。

有关更多信息和示例语法，请参阅 putbucKetenCryption 。

使用适用于 Java 的 AWS 软件开发工具包

以下示例使用 AWS SDK for Java，通过 SSE-KMS 和 S3 存储桶密钥启用默认存储桶加密。

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

ServerSideEncryptionByDefault serverSideEncryptionByDefault = new
 ServerSideEncryptionByDefault()
    .withSSEAlgorithm(SSEAlgorithm.KMS);
ServerSideEncryptionRule rule = new ServerSideEncryptionRule()
    .withApplyServerSideEncryptionByDefault(serverSideEncryptionByDefault)
    .withBucketKeyEnabled(true);
ServerSideEncryptionConfiguration serverSideEncryptionConfiguration =
    new ServerSideEncryptionConfiguration().withRules(Collections.singleton(rule));

SetBucketEncryptionRequest setBucketEncryptionRequest = new
 SetBucketEncryptionRequest()
    .withServerSideEncryptionConfiguration(serverSideEncryptionConfiguration)
    .withBucketName(bucketName);

s3client.setBucketEncryption(setBucketEncryptionRequest);
```

使用 AWS CLI

以下示例使用 AWS CLI，通过 SSE-KMS 和 S3 存储桶密钥启用默认存储桶加密。将 *user input placeholders* 替换为您自己的信息。

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

使用 AWS CloudFormation

有关使用 AWS CloudFormation 配置 S3 存储桶密钥的更多信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::S3::Bucket ServerSideEncryptionRule。

在对象级别配置 S3 存储桶密钥

当您使用 REST API、AWS SDK 或 AWS CLI 执行 PUT 或 COPY 操作时，您可以通过添加带有 `true` 或 `false` 值的 `x-amz-server-side-encryption-bucket-key-enabled` 请求标头在对象级别启用或禁用 S3 存储桶密钥。S3 存储桶密钥通过减少从 Amazon S3 到 AWS KMS 的请求流量，降低了使用 AWS Key Management Service（AWS KMS）（SSE-KMS）进行服务器端加密的成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

当您使用 PUT 或 COPY 操作为对象配置 S3 存储桶密钥时，Amazon S3 仅更新该对象的设置。目标存储桶的 S3 存储桶密钥设置不会更改。如果您在启用了 S3 存储桶密钥的存储桶中提交对于 KMS 加密对象的 PUT 或 COPY 请求，则除非您禁用了请求标头中的密钥，否则您的对象级操作将自动使用 S3 存储桶密钥。如果您未为对象指定 S3 存储桶密钥，则 Amazon S3 会将目标存储桶的 S3 存储桶密钥设置应用于该对象。

先决条件：

在将对象配置为使用 S3 存储桶密钥之前，请查看 启用 S3 存储桶密钥之前需要注意的更改。

主题

- Amazon S3 批量操作
- 使用 REST API
- 使用适用于 Java 的 AWS SDK（PutObject）
- 使用 AWS CLI (PutObject)

Amazon S3 批量操作

要加密现有 Amazon S3 对象，可以使用 Amazon S3 批量操作。您为 S3 批量操作提供了要操作的对象列表，而批量操作调用相应的 API 来执行指定的操作。

您可以使用 S3 批量操作复制操作复制现有的未加密对象，并将其作为加密对象写回同一存储桶。单个批量操作作业可对数十亿个对象执行指定操作。有关更多信息，请参阅 对 Amazon S3 对象执行大规模批量操作 和 使用 Amazon S3 批量操作加密对象。

使用 REST API

使用 SSE-KMS 时，您可以使用以下 API 操作为对象启用 S3 存储桶密钥：

- PutObject – 上传对象时，您可以指定 `x-amz-server-side-encryption-bucket-key-enabled` 请求标头以在对象级别启用或禁用 S3 存储桶密钥。
- CopyObject – 当您复制对象并配置 SSE-KMS 时，您可以指定 `x-amz-server-side-encryption-bucket-key-enabled` 请求标头以为对象启用或禁用 S3 存储桶密钥。
- POST 对象 – 当您使用 POST 操作上传对象并配置 SSE-KMS 时，您可以使用 `x-amz-server-side-encryption-bucket-key-enabled` 表单字段为对象启用或禁用 S3 存储桶密钥。
- CreateMultipartUpload – 当您使用 `CreateMultipartUpload` API 操作上传大型对象并配置 SSE-KMS 时，您可以使用 `x-amz-server-side-encryption-bucket-key-enabled` 请求标头为对象启用或禁用 S3 存储桶密钥。

要在对象级别启用 S3 存储桶密钥，请包含 `x-amz-server-side-encryption-bucket-key-enabled` 请求标头。有关 SSE-KMS 和 REST API 的更多信息，请参阅 使用 REST API。

使用适用于 Java 的 AWS SDK（PutObject）

您可以使用以下示例通过 AWS SDK for Java 在对象级别配置 S3 存储桶密钥。

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

String bucketName = "amzn-s3-demo-bucket1";
String keyName = "key name for object";
String contents = "file contents";

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName,
 contents)
    .withBucketKeyEnabled(true);

s3client.putObject(putObjectRequest);
```

使用 AWS CLI (PutObject)

您可以使用以下 AWS CLI 示例作为 `PutObject` 请求的一部分在对象级别配置 S3 存储桶密钥。

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key object key name --server-side-encryption aws:kms --bucket-key-enabled --body filepath
```

查看 S3 存储桶密钥的设置

您可以使用 Amazon S3 控制台、REST API、AWS Command Line Interface（AWS CLI）或 AWS SDK 在存储桶或对象级别查看 S3 存储桶密钥的设置。

S3 存储桶密钥减少了从 Amazon S3 到 AWS KMS 的请求流量，从而降低了使用 AWS Key Management Service（SSE-KMS）进行服务器端加密的成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

要查看存储桶或已从存储桶配置继承了 S3 存储桶密钥设置的对象的 S3 存储桶密钥设置，您需要获得执行 `s3:GetEncryptionConfiguration` 操作的权限。有关更多信息，请参阅 Amazon Simple Storage Service API 参考中的 GetBucketEncryption。

使用 S3 控制台

在 S3 控制台中，您可以查看存储桶或对象的 S3 存储桶密钥设置。S3 存储桶密钥设置继承自存储桶配置，除非源对象已配置 S3 存储桶密钥。

同一存储桶中的对象和文件夹可以具有不同的 S3 存储桶密钥设置。例如，如果您使用 REST API 上传对象并为该对象启用了 S3 存储桶密钥，则即使在目标存储桶中禁用了 S3 存储桶密钥，该对象仍会在目标存储桶中保留其 S3 存储桶密钥设置。另一个示例是，如果您为现有存储桶启用 S3 存储桶密钥，则存储桶中已存在的对象将不使用 S3 存储桶密钥。但是，新对象已启用 S3 存储桶密钥。

查看存储桶的 S3 存储桶密钥设置

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（存储桶）列表中，请选择要为其启用 S3 存储桶密钥的存储桶。
4. 选择 Properties (属性)。
5. 在默认加密部分的存储桶密钥下，您可以看到存储桶的 S3 存储桶密钥设置。

如果您看不到 S3 存储桶密钥设置，则可能没有执行 `s3:GetEncryptionConfiguration` 操作的权限。有关更多信息，请参阅 Amazon Simple Storage Service API 参考 中的 GetBucketEncryption。

查看对象的 S3 存储桶密钥设置

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要为其启用 S3 存储桶密钥的存储桶。
3. 在 Objects（对象）列表中，请选择对象名称。
4. 在 Details (详细信息) 选项卡上，Server-side encryption settings (服务器端加密设置) 下，请选择 Edit (编辑)。

   在存储桶密钥下，您可以看到对象的 S3 存储桶密钥设置。您无法编辑此设置。

使用 AWS CLI

返回存储桶级别 S3 存储桶密钥设置

要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 get-bucket-encryption。

返回对象级别 S3 存储桶密钥设置

要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。

```
aws s3api head-object --bucket amzn-s3-demo-bucket1 --key my_images.tar.bz2
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 head-object。

使用 REST API

返回存储桶级别 S3 存储桶密钥设置

要返回存储桶的加密信息，包括 S3 存储桶密钥的设置，请使用 `GetBucketEncryption` 操作。S3 存储桶密钥设置将在 `ServerSideEncryptionConfiguration` 元素中的响应正文

中与 BucketKeyEnabled 设置一起返回。有关更多信息，请参阅 Amazon S3 API 参考中的 GetBucketEncryption。

返回 S3 存储桶密钥的对象级别设置

要返回对象的 S3 存储桶密钥状态，请使用 HeadObject 操作。HeadObject 返回 x-amz-server-side-encryption-bucket-key-enabled 响应标头，以显示是否为对象启用或禁用了 S3 存储桶密钥。有关更多信息，请参阅 Amazon S3 API 参考中的 HeadObject。

如果为对象配置了 S3 存储桶密钥，则以下 API 操作还会返回 x-amz-server-side-encryption-bucket-key-enabled 响应标头：

- PutObject
- PostObject
- CopyObject
- CreateMultipartUpload
- UploadPartCopy
- UploadPart
- CompleteMultipartUpload
- GetObject

使用具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）

在将对象上传到 Amazon S3 时，使用具有 AWS Key Management Service（AWS KMS）密钥的双层服务器端加密（DSSE-KMS）将会对于对象应用两层加密。DSSE-KMS 可帮助您更轻松地满足合规性标准，这些标准要求您对数据应用多层加密并完全控制您的加密密钥。

将 DSSE-KMS 加密用于 Amazon S3 存储桶时，AWS KMS 密钥必须位于该存储桶所在的同一区域中。此外，当为对象请求 DSSE-KMS 时，作为对象元数据一部分的 S3 校验和将以加密形式存储。有关校验和的更多信息，请参阅检查对象完整性。

使用 DSSE-KMS 和 AWS KMS keys 无需支付额外费用。有关 DSSE-KMS 定价的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS key 概念以及 AWS KMS 定价。

> **Note**
>
> DSSE-KMS 不支持 S3 存储桶密钥。

要求使用具有 AWS KMS keys 的双层服务器端加密（DSSE-KMS）

如果要求对特定 Amazon S3 存储桶中的所有对象进行双层服务器端加密，则可以使用存储桶策略。例如，如果请求不包含用于请求服务器端加密（DSSE-KMS）的 `x-amz-server-side-encryption` 标头，则下面的存储桶策略将拒绝所有人的上传对象（`s3:PutObject`）权限。

```
{
            "Version":"2012-10-17",
            "Id":"PutObjectPolicy",
            "Statement":[{
                  "Sid":"DenyUnEncryptedObjectUploads",
                  "Effect":"Deny",
                  "Principal":"*",
                  "Action":"s3:PutObject",
                  "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*",
                  "Condition":{
                     "StringNotEquals":{
                        "s3:x-amz-server-side-encryption":"aws:kms:dsse"
                     }
                  }
               }
            ]
         }
```

主题

- 指定具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）

指定具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

默认情况下，所有 Amazon S3 存储桶都配置了加密，所有上传到 S3 存储桶的新对象都会自动静态加密。具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）是 Amazon S3 中每个存储桶的默认加密配

置。要使用其他类型的加密，您可以指定要在 S3 PUT 请求中使用的服务器端加密类型，也可以在目标存储桶中设置默认加密配置。

如果您想在 PUT 请求中指定不同的加密类型，则可以使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）或具有客户提供的密钥的服务器端加密（SSE-C）。如果您想在目标存储桶中设置不同的默认加密配置，则可以使用 SSE-KMS 或 DSSE-KMS。

当您上传新对象或复制现有对象时，您可以应用加密。

您可以使用 Amazon S3 控制台、Amazon S3 REST API 和 AWS Command Line Interface（AWS CLI）指定 DSSE-KMS。有关更多信息，请参阅以下主题。

> **Note**
>
> 您可以在 Amazon S3 中使用多区域 AWS KMS keys。但是，Amazon S3 目前将多区域密钥视为单区域密钥，且不使用密钥的多区域特征。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的使用多区域密钥。

> **Note**
>
> 如果您希望使用其他账户拥有的 KMS 密钥，则您必须有权使用该密钥。有关 KMS 密钥的跨账户权限的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建其他账户可以使用的 KMS 密钥。

使用 S3 控制台

本节介绍如何使用 Amazon S3 控制台，将对象的加密类型设置或更改为使用具有 AWS Key Management Service（AWS KMS）密钥的双层服务器端加密（DSSE-KMS）。

> **Note**
>
> - 如果更改对象的加密方法，则会创建一个新对象来替换旧对象。如果启用 S3 版本控制，则会创建对象的新版本，而现有对象将变为旧版本。更改属性的角色也会成为新对象（或对象版本）的拥有者。

- 如果您要更改具有用户定义标签的对象的加密类型，您必须拥有 `s3:GetObjectTagging` 权限。如果您要更改没有用户定义标签但大小超过 16 MB 的对象的加密类型，您还必须拥有 `s3:GetObjectTagging` 权限。

  如果目标存储桶策略拒绝 `s3:GetObjectTagging` 操作，则将更新对象的加密类型，但将从对象中移除用户定义的标签，并且您将收到错误。

添加或更改对象的加密

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择包含您想要加密的对象的存储桶的名称。
4. 在对象列表中，选择要为其添加或更改加密的对象旁边的复选框。

   将显示对象的详细信息页面，其中有几个部分显示您的对象的属性。
5. 选择属性选项卡。
6. 向下滚动到默认加密部分，然后选择编辑。

   编辑默认加密页面随即打开。
7. 在加密类型下，选择具有 AWS Key Management Service 密钥的双层服务器端加密（DSSE-KMS）。
8. 在 AWS KMS 密钥下，执行以下操作以选择您的 KMS 密钥：

   - 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 中进行选择，然后从可用密钥的列表中选择您的 KMS 密钥。

     AWS 托管式密钥（`aws/s3`）和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。
   - 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
   - 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

     有关创建 AWS KMS key 的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

> **Important**
>
> 您只能使用与存储桶所在相同的 AWS 区域中可用的 KMS 密钥。Amazon S3 控制台仅列出与存储桶位于同一区域中的前 100 个 KMS 密钥。要使用未列出的 KMS 密钥，您必须输入 KMS 密钥 ARN。如果您希望使用其他账户拥有的 KMS 密钥，则必须首先有权使用该密钥，然后必须输入相应的 KMS 密钥 ARN。
>
> Amazon S3 仅支持对称加密 KMS 密钥，不支持非对称 KMS 密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的确定非对称 KMS 密钥。

9. 对于存储桶密钥，选择禁用。DSSE-KMS 不支持 S3 存储桶密钥。
10. 选择 Save Changes（保存更改）。

> **Note**
>
> 此操作将加密应用于所有指定的对象。加密文件夹时，请等待保存操作完成，然后再将新对象添加到文件夹。

使用 REST API

创建对象时（即上传新对象或复制现有对象时），您可以指定使用具有 AWS KMS keys 的双层服务器端加密（DSSE-KMS）来加密数据。为此，请将 `x-amz-server-side-encryption` 标头添加到请求。将标头的值设置为加密算法 `aws:kms:dsse`。Amazon S3 通过返回响应标头 `x-amz-server-side-encryption` 来确认已使用 DSSE-KMS 加密来存储对象。

如果您指定值为 `aws:kms:dsse` 的 `x-amz-server-side-encryption` 标头，则还可以使用以下请求标头：

- `x-amz-server-side-encryption-aws-kms-key-id: SSEKMSKeyId`
- `x-amz-server-side-encryption-context: SSEKMSEncryptionContext`

主题

- 支持 DSSE-KMS 的 Amazon S3 REST API 操作
- 加密上下文（x-amz-server-side-encryption-context）
- AWS KMS 密钥 ID（x-amz-server-side-encryption-aws-kms-key-id）

支持 DSSE-KMS 的 Amazon S3 REST API 操作

以下 REST API 操作接受 `x-amz-server-side-encryption`、`x-amz-server-side-encryption-aws-kms-key-id` 和 `x-amz-server-side-encryption-context` 请求标头。

- PutObject – 使用 PUT API 操作上传数据时，您可以指定这些请求标头。
- CopyObject – 复制对象时，您同时具有源对象和目标对象。如果使用 `CopyObject` 操作传递 DSSE-KMS 标头，它们仅应用于目标对象。复制现有对象时，不论源对象是否已经加密，都不会加密目标对象，除非您显式请求服务器端加密。
- POST Object – 使用 `POST` 操作上传对象时，可在表单字段（而不是在请求标头）中提供相同的信息。
- CreateMultipartUpload – 使用分段上传来上传大型对象时，可以在 `CreateMultipartUpload` 请求中指定这些标头。

使用服务器端加密存储对象时，以下 REST API 操作的响应标头将返回 `x-amz-server-side-encryption` 标头。

- PutObject
- CopyObject
- POST 对象
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

> **Important**
>
> - 如果您不让使用安全套接字层（SSL）、传输层安全性协议（TLS）或签名版本 4 发出针对受 AWS KMS 保护的对象的所有 `GET` 和 `PUT` 请求，则这些请求都将失败。
> - 如果对象使用 DSSE-KMS，则不应对 `GET` 请求和 `HEAD` 请求发送加密请求标头，否则您将得到 HTTP 400（错误请求）错误。

加密上下文（**x-amz-server-side-encryption-context**）

如果您指定 x-amz-server-side-encryption:aws:kms:dsse，Amazon S3 API 将支持带有 x-amz-server-side-encryption-context 标头的加密上下文。加密上下文是一组键值对，其中包含有关数据的其他上下文信息。

Amazon S3 自动使用对象的 Amazon 资源名称（ARN）作为加密上下文对；例如 arn:aws:s3:::*object_ARN*。

您可以选择使用 x-amz-server-side-encryption-context 标头提供其他的加密上下文对。但是，由于加密上下文未加密，请确保它不包含敏感信息。Amazon S3 将此额外的密钥对与默认加密上下文一起存储。

有关 Amazon S3 中加密上下文的信息，请参阅 加密上下文。有关加密上下文的一般信息，请参阅 AWS Key Management Service 开发人员指南中的 AWS Key Management Service 概念 - 加密上下文。

AWS KMS 密钥 ID（**x-amz-server-side-encryption-aws-kms-key-id**）

您可以使用 x-amz-server-side-encryption-aws-kms-key-id 标头指定用于保护数据的客户自主管理型密钥的 ID。如果您指定 x-amz-server-side-encryption:aws:kms:dsse 标头但未提供 x-amz-server-side-encryption-aws-kms-key-id 标头，Amazon S3 将使用 AWS 托管式密钥（aws/s3）来保护数据。如果要使用客户托管密钥，则必须提供客户托管密钥的 x-amz-server-side-encryption-aws-kms-key-id 标头。

> **Important**
>
> 在 Amazon S3 中使用 AWS KMS key 进行服务器端加密时，您必须选择对称加密 KMS 密钥。Amazon S3 仅支持对称加密 KMS 密钥。有关这些密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的对称加密 KMS 密钥。

使用 AWS CLI

当您上传新对象或复制现有对象时，可以指定使用 DSSE-KMS 来加密数据。为此，请将 --server-side-encryption aws:kms:dsse 参数添加到请求。使用 --ssekms-key-id *example-key-id* 参数添加您创建的客户自主管理型 AWS KMS 密钥。如果您指定 --server-side-encryption aws:kms:dsse，但未提供 AWS KMS 密钥 ID，则 Amazon S3 将使用 AWS 托管式密钥（aws/s3）。

```
aws s3api put-object --bucket DOC-EXAMPLE-BUCKET --key example-object-key --server-
side-encryption aws:kms:dsse --ssekms-key-id example-key-id --body filepath
```

您可以通过将未加密的对象复制回原位来加密该对象以使用 DSSE-KMS。

```
aws s3api copy-object --bucket DOC-EXAMPLE-BUCKET --key example-object-key --
body filepath --bucket DOC-EXAMPLE-BUCKET --key example-object-key --sse aws:kms:dsse
 --sse-kms-key-id example-key-id --body filepath
```

使用具有客户提供的密钥的服务器端加密（SSE-C）

服务器端加密是为了保护静态数据。服务器端加密仅加密对象数据而非加密对象元数据。使用具有客户提供的密钥的服务器端加密（SSE-C），您可以存储使用自己的加密密钥加密的数据。使用您作为请求的一部分提供的加密密钥，Amazon S3 在其写入磁盘时管理数据加密，并在您访问对象时管理数据解密。因此，您不需要维护任何代码来执行数据加密和解密。您只需管理您提供的加密密钥。

在您上传对象时，Amazon S3 将使用您提供的加密密钥对您的数据应用 AES-256 加密。然后，Amazon S3 从内存中删除此加密密钥。在检索对象时，必须提供相同的加密密钥作为您请求的一部分。Amazon S3 在将对象数据返回给您之前，会首先验证您提供的加密密钥是否匹配，然后再解密对象。

使用 SSE-C 没有额外费用。但是，配置和使用 SSE-C 的请求会产生标准的 Amazon S3 请求费用。有关定价的信息，请参阅 Amazon S3 定价。

> **Note**
>
> Amazon S3 不存储您提供的加密密钥，而是存储加密密钥的添加了随机数据的 HMAC 散列消息认证码（HMAC）值，以验证将来的请求。无法使用添加了随机数据的 HMAC 值来推导出加密密钥的值或解密加密对象的内容。这意味着，如果您丢失加密密钥，则会失去该对象。

S3 复制支持使用 SSE-C 加密的对象。有关复制加密对象的更多信息，请参阅the section called “复制加密对象”。

有关 SSE-C 的更多信息，请参阅以下主题。

主题

- SSE-C 概览

- 要求和限制 SSE-C
- 预签名 URL 和 SSE-C
- 指定使用客户提供的密钥的服务器端加密 (SSE-C)。

SSE-C 概览

本部分提供 SSE-C 的概述。使用 SSE-C 时，请记住以下注意事项。

- 您必须使用 HTTPS。

> **Important**
>
> 在使用 SSE-C 时，Amazon S3 会拒绝通过 HTTP 发出的所有请求。出于安全原因，我们建议您考虑您错误地通过 HTTP 发送的任何密钥都会遭泄露。丢弃该密钥，并根据需要轮换密钥。

- 响应中的实体标签（ETag）不是对象数据的 MD5 哈希。
- 您管理哪个加密密钥用于加密哪个对象的映射。Amazon S3 不存储加密密钥。您负责跟踪为哪个对象提供了哪个加密密钥。
  - 如果您的存储桶启用了版本控制，则您使用此特征上传的每个对象版本可能都具有自己的加密密钥。您负责跟踪哪个加密密钥用于哪个对象版本。
  - 因为您在客户端管理加密密钥，所以也要在客户端管理所有额外的保护措施，例如密钥轮换。

> **Warning**
>
> 如果您丢失加密密钥，则在没有加密密钥的情况下对于对象的任何 GET 请求都会失败，并且您将丢失该对象。

要求和限制 SSE-C

如果要求对特定 Amazon S3 存储桶中的所有对象执行 SSE-C，请使用存储桶策略。

例如，如果请求不包括用于请求 SSE-C 的 `x-amz-server-side-encryption-customer-algorithm` 标头，以下存储桶策略将拒绝针对所有此类请求的上传对象（`s3:PutObject`）权限。

```
{
```

```
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
            "Sid": "RequireSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"
                }
            }
        }
    ]
}
```

也可以使用策略，以限制对特定 Amazon S3 存储桶中的所有对象进行服务器端加密。例如，如果请求包含用于请求 SSE-C 的 `x-amz-server-side-encryption-customer-algorithm` 标头，则下面的存储桶策略将对所有人拒绝上传对象（`s3:PutObject`）权限。

```
{
    "Version": "2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
            "Sid": "RestrictSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "false"
                }
            }
        }
    ]
}
```

> **Important**
>
> 如果您使用存储桶策略在 `s3:PutObject` 上请求 SSE-C，则必须在所有分段上传请求中（CreateMultipartUpload、UploadPart 和 CompleteMultipartUpload）包括 `x-amz-server-side-encryption-customer-algorithm` 标头。

预签名 URL 和 SSE-C

您可以生成可用于上传新对象、检索现有对象或检索对象元数据等操作的预签名 URL。预签名 URL 支持 SSE-C，如下所示：

- 在创建预签名 URL 时，您必须在签名计算中使用 `x-amz-server-side-encryption-customer-algorithm` 标头指定算法。
- 在使用预签名 URL 上传新对象、检索现有对象或仅检索对象元数据时，您必须在您的客户端应用程序的请求中提供所有加密标头。

> **Note**
>
> 对于非 SSE-C 对象，您可以生成预签名 URL，并将该 URL 直接复制到浏览器中以访问数据。
> 但是，不能对 SSE-C 对象执行此操作，因为除了预签名 URL 外，还必须包含 SSE-C 对象特定的 HTTP 标头。因此，您只能以编程方式将预签名 URL 用于 SSE-C 对象。

有关预签名 URL 的更多信息，请参阅the section called “使用预签名 URL”。

指定使用客户提供的密钥的服务器端加密 (SSE-C)。

在使用 REST API 创建对象时，您可以使用客户提供的密钥（SSE-C）指定服务器端加密。使用 SSE-C 时，必须使用以下请求标头提供加密密钥信息。

名称	描述
`x-amz-server-side-encryption-customer-algorithm`	使用此标头来指定加密算法。标头值必须为 AES256。

名称	描述
`x-amz-server-side-encryption-customer-key`	使用此标头来提供 256 位的 base64 编码的加密密钥以供 Amazon S3 用于加密或解密您的数据。
`x-amz-server-side-encryption-customer-key-MD5`	使用此标头根据 RFC 1321 提供加密密钥的 base64 编码的 128 位 MD5 摘要。Amazon S3 使用此标头进行消息完整性检查以确保加密密钥的传输无误。

您可以使用 AWS SDK 包装库将这些标头添加到您的请求中。如果需要，您可以直接在应用程序中调用 Amazon S3 REST API。

> **Note**
>
> 您不能使用 Amazon S3 控制台上传对象并请求 SSE-C。也不能使用控制台来更新使用 SSE-C 存储的现有对象（例如，更改存储类或添加元数据）。

使用 REST API

支持 SSE-C 的 Amazon S3 REST API

以下 Amazon S3 API 支持使用客户提供的加密密钥进行服务器端加密 (SSE-C)。

- GET 操作 – 在使用 GET API 检索对象（请参阅 GET Object）时，您可以指定请求标头。
- HEAD 操作 – 要使用 HEAD API 检索对象元数据（请参阅 HEAD Object），可以指定这些请求标头。
- PUT 操作 – 使用 PUT Object API 上传数据（请参阅 PUT Object）时，可以指定这些请求标头。
- 分段上传 – 在使用分段上传 API 上传大对象时，可以指定这些标头。您可以在以下请求中指定这些标头：启动请求（请参阅启动分段上传）和每个后续分段上传请求（请参阅上传分段或上传分段 - 复制）。对于每个分段上传请求，加密信息必须与您在启动分段上传请求中提供的信息相同。
- POST 操作 – 使用 POST 操作上传对象（请参阅 POST 对象）时，可在表单字段而不是请求标头中提供相同的信息。
- 复制操作 – 复制对象（请参阅 PUT Object - 复制）时，您同时具有源对象和目标对象：

- 如果您希望使用具有 AWS 托管式密钥的服务器端加密对目标对象加密，则必须提供 `x-amz-server-side-encryption` 请求标头。
- 如果您希望使用 SSE-C 对目标对象加密，则必须使用上表中描述的三个标头提供加密信息。
- 如果源对象是使用 SSE-C 加密的，则您必须使用以下标头提供加密密钥信息，以便 Amazon S3 可以解密对象以进行复制。

名称	描述
`x-amz-copy-source-server-side-encryption-customer-algorithm`	包括此标头以指定 Amazon S3 用于解密源对象的算法。此值必须是 AES256。
`x-amz-copy-source-server-side-encryption-customer-key`	包括此标头以提供 base64 编码的加密密钥，供 Amazon S3 用于解密源对象。此加密密钥必须是您在创建源对象时为 Amazon S3 提供的加密密钥。否则，Amazon S3 无法解密对象。
`x-amz-copy-source-server-side-encryption-customer-key-MD5`	包括此标头以根据 RFC 1321 提供加密密钥的 base64 编码的 128 位 MD5 摘要。

使用 AWS SDK 为 PUT、GET、Head 和 Copy 操作指定 SSE-C

以下示例演示如何为对象请求使用客户提供的密钥的服务器端加密 (SSE-C)。这些示例执行以下操作。每个操作均演示了如何在请求中指定 SSE-C 相关标头：

- 放置对象 – 上传对象，并请求使用客户提供的加密密钥的服务器端加密。
- 获取对象 – 下载上一步中上传的对象。在请求中，应提供上传对象时提供的同一加密信息。Amazon S3 需要此信息来解密对象，以便将对象返回给您。
- 获取对象元数据 – 检索对象的元数据。提供创建对象时使用的同一加密信息。

- 复制对象 – 复制之前上传的对象的副本。因为源对象是使用 SSE-C 存储的，因此必须在复制请求中提供其加密信息。默认情况下，仅当您显式请求加密时，Amazon S3 才会为对象的副本加密。此示例指示 Amazon S3 存储对象的加密副本。

Java

> **Note**
>
> 本示例显示如何在单个操作中上传对象。当使用分段上传 API 上传大型对象时，应按照此示例中所示的方式提供加密信息。有关使用 AWS SDK for Java 的分段上传的示例，请参阅使用分段上传操作上传对象。

要添加必需的加密信息，请在请求中包含 `SSECustomerKey`。有关该 `SSECustomerKey` 课程的更多信息，请参阅 REST API 部分。

有关 SSE-C 的信息，请参阅 使用具有客户提供的密钥的服务器端加密（SSE-C）。有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import javax.crypto.KeyGenerator;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class ServerSideEncryptionUsingClientSideEncryptionKey {
    private static SSECustomerKey SSE_KEY;
    private static AmazonS3 S3_CLIENT;
```

```
   private static KeyGenerator KEY_GENERATOR;

   public static void main(String[] args) throws IOException,
NoSuchAlgorithmException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String uploadFileName = "*** File path ***";
        String targetKeyName = "*** Target key name ***";

        // Create an encryption key.
        KEY_GENERATOR = KeyGenerator.getInstance("AES");
        KEY_GENERATOR.init(256, new SecureRandom());
        SSE_KEY = new SSECustomerKey(KEY_GENERATOR.generateKey());

        try {
            S3_CLIENT = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Upload an object.
            uploadObject(bucketName, keyName, new File(uploadFileName));

            // Download the object.
            downloadObject(bucketName, keyName);

            // Verify that the object is properly encrypted by attempting to
retrieve it
            // using the encryption key.
            retrieveObjectMetadata(bucketName, keyName);

            // Copy the object into a new object that also uses SSE-C.
            copyObject(bucketName, keyName, targetKeyName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
   }
```

```
   private static void uploadObject(String bucketName, String keyName, File file) {
       PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
file).withSSECustomerKey(SSE_KEY);
       S3_CLIENT.putObject(putRequest);
       System.out.println("Object uploaded");
   }

   private static void downloadObject(String bucketName, String keyName) throws
IOException {
       GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName,
keyName).withSSECustomerKey(SSE_KEY);
       S3Object object = S3_CLIENT.getObject(getObjectRequest);

       System.out.println("Object content: ");
       displayTextInputStream(object.getObjectContent());
   }

   private static void retrieveObjectMetadata(String bucketName, String keyName) {
       GetObjectMetadataRequest getMetadataRequest = new
GetObjectMetadataRequest(bucketName, keyName)
               .withSSECustomerKey(SSE_KEY);
       ObjectMetadata objectMetadata =
S3_CLIENT.getObjectMetadata(getMetadataRequest);
       System.out.println("Metadata retrieved. Object size: " +
objectMetadata.getContentLength());
   }

   private static void copyObject(String bucketName, String keyName, String
targetKeyName)
           throws NoSuchAlgorithmException {
       // Create a new encryption key for target so that the target is saved using
       // SSE-C.
       SSECustomerKey newSSEKey = new SSECustomerKey(KEY_GENERATOR.generateKey());

       CopyObjectRequest copyRequest = new CopyObjectRequest(bucketName, keyName,
bucketName, targetKeyName)
               .withSourceSSECustomerKey(SSE_KEY)
               .withDestinationSSECustomerKey(newSSEKey);

       S3_CLIENT.copyObject(copyRequest);
       System.out.println("Object copied");
   }
```

```
    private static void displayTextInputStream(S3ObjectInputStream input) throws
 IOException {
        // Read one line at a time from the input stream and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

.NET

> **Note**
>
> 有关使用分段上传 API 上传大型对象的示例，请参阅 使用分段上传操作上传对象 和 使用 AWS SDK（低级别 API）。

有关 SSE-C 的信息，请参阅 使用具有客户提供的密钥的服务器端加密（SSE-C）。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSEClientEncryptionKeyObjectOperationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for new object created ***";
        private const string copyTargetKeyName = "*** key name for object copy ***";
        // Specify your bucket region (an example region is shown).
```

```
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            ObjectOpsUsingClientEncryptionKeyAsync().Wait();
        }
        private static async Task ObjectOpsUsingClientEncryptionKeyAsync()
        {
            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // 1. Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(base64Key);
                // 2. Download the object and verify that its contents matches what
you uploaded.
                await DownloadObjectAsync(base64Key, putObjectRequest);
                // 3. Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(base64Key);
                // 4. Copy both the source and target objects using server-side
encryption with
                //    a customer-provided encryption key.
                await CopyObjectAsync(aesEncryption, base64Key);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }
```

```
        private static async Task<PutObjectRequest> UploadObjectAsync(string
base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };
            PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }
        private static async Task DownloadObjectAsync(string base64Key,
PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (String.Compare(putObjectRequest.ContentBody, content) == 0)
                    Console.WriteLine("Object content is same as we uploaded");
                else
                    Console.WriteLine("Error...Object content is not same.");

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
```

```
                    Console.WriteLine("Object encryption method is AES256, same as
we set");
                else
                    Console.WriteLine("Error...Object encryption method is not the
same as AES256 we set");

                // Assert.AreEqual(putObjectRequest.ContentBody, content);
                // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getResponse.ServerSideEncryptionCustomerMethod);
            }
        }
        private static async Task GetObjectMetadataAsync(string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used is:
{0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
            // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }
        private static async Task CopyObjectAsync(Aes aesEncryption, string
base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
```

```
                DestinationKey = copyTargetKeyName,
                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,
                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
}
```

使用 AWS SDK 为分段上传指定 SSE-C

上一部分中的示例显示了如何在 PUT、GET、Head 和 Copy 操作中请求使用客户提供的密钥的服务器端加密 (SSE-C)。本节介绍支持 SSE-C 的其他 Amazon S3 API。

Java

要上传大型对象，您可以使用分段上传 API（请参阅使用分段上传来上传和复制对象）。可以使用高级或低级 API 上传大型对象。这些 API 支持在请求中使用与加密相关的标头。

- 当使用高级别 TransferManager API 时，应在 PutObjectRequest 中提供特定于加密的标头 (请参阅 使用分段上传操作上传对象)。
- 当使用低级别 API 时，应提供 InitiateMultipartUploadRequest 中的加密相关信息，后跟每个 UploadPartRequest 中的相同加密信息。不需要在 CompleteMultipartUploadRequest 中提供任何特定于加密的标头。有关示例，请参阅使用 AWS SDK（低级别 API）。

以下示例使用 TransferManager 创建对象并显示如何提供 SSE-C 相关信息。本示例执行以下操作：

- 使用 TransferManager.upload() 方法创建对象。在 PutObjectRequest 实例中，应提供要请求的加密密钥信息。Amazon S3 将使用客户提供的密钥对于对象进行加密。
- 通过调用 TransferManager.copy() 方法创建对象的副本。该示例指示 Amazon S3 使用新的 SSECustomerKey 对对象副本进行加密。由于源对象使用 SSE-C 加密，因此

CopyObjectRequest 还提供了源对象的加密密钥，以便 Amazon S3 可以在复制对象之前解密对象。

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.SSECustomerKey;
import com.amazonaws.services.s3.transfer.Copy;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import javax.crypto.KeyGenerator;
import java.io.File;
import java.security.SecureRandom;

public class ServerSideEncryptionCopyObjectUsingHLwithSSEC {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String fileToUpload = "*** File path ***";
        String keyName = "*** New object key name ***";
        String targetKeyName = "*** Key name for object copy ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
                    .withS3Client(s3Client)
                    .build();

            // Create an object from a file.
```

```
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
keyName, new File(fileToUpload));

            // Create an encryption key.
            KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
            keyGenerator.init(256, new SecureRandom());
            SSECustomerKey sseCustomerEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());

            // Upload the object. TransferManager uploads asynchronously, so this
call
            // returns immediately.
            putObjectRequest.setSSECustomerKey(sseCustomerEncryptionKey);
            Upload upload = tm.upload(putObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object created.");

            // Copy the object and store the copy using SSE-C with a new key.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucketName,
keyName, bucketName, targetKeyName);
            SSECustomerKey sseTargetObjectEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());
            copyObjectRequest.setSourceSSECustomerKey(sseCustomerEncryptionKey);

copyObjectRequest.setDestinationSSECustomerKey(sseTargetObjectEncryptionKey);

            // Copy the object. TransferManager copies asynchronously, so this call
returns
            // immediately.
            Copy copy = tm.copy(copyObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            copy.waitForCompletion();
            System.out.println("Copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        }
    }
}
```

.NET

要上传大型对象，您可以使用分段上传 API（请参阅使用分段上传来上传和复制对象）。AWS适用于 .NET 的 SDK 提供了高级或低级 API 来上传大型对象。这些 API 支持在请求中使用与加密相关的标头。

- 当使用高级 `Transfer-Utility` API 时，可以在 `TransferUtilityUploadRequest` 中提供特定于加密的标头，如下所示。有关代码示例，请参阅 使用分段上传操作上传对象。

```
TransferUtilityUploadRequest request = new TransferUtilityUploadRequest()
{
    FilePath = filePath,
    BucketName = existingBucketName,
    Key = keyName,
    // Provide encryption information.
    ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key,
};
```

- 当使用低级 API 时，可以在启动分段上传请求中提供加密相关的信息，并在后续的分段上传请求中提供相同加密信息。不需要在完成的分段上传请求中提供任何特定于加密的标头。有关示例，请参阅使用 AWS SDK（低级别 API）。

  下面是一个低级分段上传示例，该示例复制一个现有大型对象。在本示例中，要复制的对象使用 SSE-C 存储在 Amazon S3 中，并且您希望使用 SSE-C 保存目标对象。在本例中，您可以执行以下操作：

  - 通过提供加密密钥和相关信息启动分段上传请求。
  - 在 `CopyPartRequest` 中提供源和目标对象加密密钥及相关信息。
  - 通过检索对象元数据获取要复制的源对象的大小。
  - 以 5 MB 大小的分段上传对象。

  Example

```
using Amazon;
```

```
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSECLowLevelMPUcopyObjectTest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string sourceKeyName      = "*** source object key name
 ***";
        private const string targetKeyName      = "*** key name for the target
 object ***";
        private const string filePath           = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CopyObjClientEncryptionKeyAsync().Wait();
        }

        private static async Task CopyObjClientEncryptionKeyAsync()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            await CreateSampleObjUsingClientEncryptionKeyAsync(base64Key,
 s3Client);

            await CopyObjectAsync(s3Client, base64Key);
        }
        private static async Task CopyObjectAsync(IAmazonS3 s3Client, string
 base64Key)
        {
            List<CopyPartResponse> uploadResponses = new List<CopyPartResponse>();
```

```
            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = targetKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB
            long firstByte = 0;
            long lastByte = partSize;

            try
            {
                // First find source object size. Because object is stored
encrypted with
                // customer provided key you need to provide encryption
information in your request.
                GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest()
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                    ServerSideEncryptionCustomerProvidedKey = base64Key // " *
**source object encryption key ***"
                };

                GetObjectMetadataResponse getObjectMetadataResponse = await
s3Client.GetObjectMetadataAsync(getObjectMetadataRequest);

                long filePosition = 0;
                for (int i = 1; filePosition <
getObjectMetadataResponse.ContentLength; i++)
                {
```

```
                    CopyPartRequest copyPartRequest = new CopyPartRequest
                    {
                        UploadId = initResponse.UploadId,
                        // Source.
                        SourceBucket = existingBucketName,
                        SourceKey = sourceKeyName,
                        // Source object is stored using SSE-C. Provide encryption
information.
                        CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        CopySourceServerSideEncryptionCustomerProvidedKey =
base64Key, //"***source object encryption key ***",
                        FirstByte = firstByte,
                        // If the last part is smaller then our normal part size
then use the remaining size.
                        LastByte = lastByte >
getObjectMetadataResponse.ContentLength ?
                            getObjectMetadataResponse.ContentLength - 1 :
lastByte,

                        // Target.
                        DestinationBucket = existingBucketName,
                        DestinationKey = targetKeyName,
                        PartNumber = i,
                        // Encryption information for the target object.
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };
                    uploadResponses.Add(await
s3Client.CopyPartAsync(copyPartRequest));
                    filePosition += partSize;
                    firstByte += partSize;
                    lastByte += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId,
                };
```

```
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId
                };
                s3Client.AbortMultipartUpload(abortMPURequest);
            }
        }
        private static async Task
CreateSampleObjUsingClientEncryptionKeyAsync(string base64Key, IAmazonS3
s3Client)
        {
            // List to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            InitiateMultipartUploadResponse initResponse =
               await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB
```

```
            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                    //PartETags = new List<PartETag>(uploadResponses)

                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);

            }
            catch (Exception exception)
            {
```

```
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
 AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId
                };
                await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
}
```

使用客户端加密保护数据

客户端加密是在本地加密数据，以帮助确保数据在传输中和静态时的安全性的行为。要在将对象发送到 Amazon S3 之前对其进行加密，请使用 Amazon S3 加密客户端。当您的对象以这种方式加密时，您的对象不会泄露给任何第三方，包括 AWS。Amazon S3 接收已经加密的对象；Amazon S3 在加密或解密对象方面不起作用。您可以使用 Amazon S3 加密客户端和服务器端加密来加密您的数据。当您向 Amazon S3 发送加密对象时，Amazon S3 无法将这些对象识别为已加密，它只会检测典型的对象。

Amazon S3 加密客户端充当您与 Amazon S3 之间的中介。在您实例化 Amazon S3 加密客户端后，您的对象将作为 Amazon S3 `PutObject` 和 `GetObject` 请求的一部分自动加密和解密。您的对象全部使用唯一的数据密钥加密。即使您将 KMS 密钥指定为包装密钥，Amazon S3 加密客户端也不会使用存储桶密钥或与存储桶密钥交互。

《Amazon S3 Encryption Client Developer Guide》侧重于 3.0 及更高版本的 Amazon S3 加密客户端。有关更多信息，请参阅《Amazon S3 加密客户端开发人员指南》中的什么是 Amazon S3 加密客户端？

有关早期版本的 Amazon S3 加密客户端的更多信息，请参阅适用于您的编程语言的《AWS SDK Developer Guide》。

- AWS SDK for Java
- AWS SDK for .NET
- AWS SDK for Go
- AWS SDK for PHP
- AWS SDK for Ruby

- AWS SDK for C++

互联网络流量隐私保护

此主题介绍 Amazon S3 如何保护从服务到其他位置的连接。

服务与本地客户端和应用之间的流量

以下连接可以和 AWS PrivateLink 合并使用，以提供私有联网和 AWS 之间的连接性：

- 一个 AWS Site-to-Site VPN 连接。有关更多信息，请参阅什么是 AWS Site-to-Site VPN？
- AWS Direct Connect 连接。有关更多信息，请参阅什么是 AWS Direct Connect？

通过网络访问 Amazon S3 是通过 AWS 发布的 API 进行的。客户端必须支持传输层安全性（TLS）1.2。我们建议使用 TLS 1.3。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Diffie-Hellman Ephemeral (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。此外，必须使用与 IAM 主体关联的访问密钥 ID 和秘密访问密钥签名请求，或者可以使用 AWS Security Token Service（STS）生成临时安全证书来签名请求。

同一区域中 AWS 资源之间的流量

Amazon S3 的 Virtual Private Cloud (VPC) 端点是 VPC 内的逻辑实体，仅允许连接到 Amazon S3。VPC 将请求路由到 Amazon S3 并将响应路由回 VPC。有关更多信息，请参阅 VPC 用户指南中的 VPC 端点。有关您可用来从 VPC 端点控制 S3 存储桶访问的示例存储桶策略，请参阅使用存储桶策略控制从 VPC 端点的访问。

AWS PrivateLink：表示 Amazon S3

借助适用于 Amazon S3 的 AWS PrivateLink，您可以在 Virtual Private Cloud (VPC) 中预置接口 VPC 端点（接口端点）。这些端点可从本地（通过 VPN 及 AWS Direct Connect）或其他 AWS 区域（通过 VPC 对等连接）中的应用程序直接访问。

接口端点由一个或多个弹性网络接口 (ENI) 代表，这些接口是从 VPC 中的子网分配的私有 IP 地址。通过接口端点向 Amazon S3 发出的请求仍留在 Amazon 网络上。您还可以通过 AWS Direct Connect 或 AWS Virtual Private Network (AWS VPN) 从本地部署应用程序访问 VPC 中的接口端点。有关如何将 VPC 与本地网络连接的更多信息，请参阅 AWS Direct Connect 用户指南和 AWS Site-to-Site VPN 用户指南。

有关接口端点的一般信息，请参阅 AWS PrivateLink 指南中的接口 VPC 端点 (AWS PrivateLink)。

主题

适用于 Amazon S3 的 VPC 端点类型

您可以使用两种类型的 VPC 端点访问 Amazon S3：网关端点和接口端点（使用 AWS PrivateLink）。网关端点是您在路由表中指定的网关，用于通过 AWS 网络从 VPC 访问 Amazon S3。接口端点通过私有 IP 地址将请求从您的 VPC 内部、本地或其他 AWS 区域中的 VPC 使用 VPC 对等连接或 AWS Transit Gateway 路由到 Amazon S3，从而扩展网关端点的功能。有关更多信息，请参阅什么是 VPC 对等连接?和 Transit Gateway 与 VPC 对等连接。

接口端点与网关端点兼容。如果您在 VPC 中有现有网关端点，则可以在同一 VPC 中使用这两种类型的端点。

适用于 Amazon S3 的网关端点	适用于 Amazon S3 的接口端点
在这两种情况下，您的网络流量仍保留在 AWS 网络中。	
使用 Amazon S3 公有 IP 地址	使用 VPC 中的私有 IP 地址访问 Amazon S3
使用相同的 Simple Storage Service (Amazon S3) DNS 名称	需要特定于端点的 Simple Storage Service (Amazon S3) DNS 名称
不允许从本地访问	允许从本地访问
不允许从其他 AWS 区域访问	允许从另一个 AWS 区域使用 VPC 对等连接或 AWS Transit Gateway 进行访问

适用于 Amazon S3 的网关端点	适用于 Amazon S3 的接口端点
不计费	已计费

有关网关端点的更多信息，请参阅 AWS PrivateLink 指南中的网关 VPC 端点。

适用于 Amazon S3 的 AWS PrivateLink 的限制和局限性

VPC 限制应用于适用于 Amazon S3 的 AWS PrivateLink。有关更多信息，请参阅《AWS PrivateLink 指南》中的接口端点注意事项和 AWS PrivateLink 限额。此外，以下限制将适用：

适用于 Amazon S3 的 AWS PrivateLink 不支持以下各项：

- 美国联邦信息处理标准 (FIPS) 端点
- 网站端点
- 传统全局端点
- S3 短划线区域端点
- Amazon S3 双堆栈端点
- 在不同 AWS 区域中的存储桶之间使用 CopyObject 或 UploadPartCopy
- 传输层安全性协议（TLS）1.1

创建 VPC 端点

要创建 VPC 接口端点，请参阅《AWS PrivateLink 指南》中的创建 VPC 端点。

访问 Amazon S3 接口端点

创建接口端点时，Amazon S3 会生成两种特定于端点的 S3 DNS 名称：区域和地区。

- 区域 DNS 名称包括唯一的 VPC 端点 ID、服务标识符、AWS 区域和以其命名的 `vpce.amazonaws.com`。例如，对于 VPC 端点 ID *`vpce-1a2b3c4d`*，生成的 DNS 名称可能类似于 *`vpce-1a2b3c4d-5e6f`*`.s3.us-east-1.vpce.amazonaws.com`。
- 区域 DNS 名称包括可用区 – 例如 *`vpce-1a2b3c4d-5e6f`*`-us-east-1a.s3.us-east-1.vpce.amazonaws.com`。如果您的架构隔离了可用区，则可以使用此选项。例如，您可以将其用于故障控制或降低区域数据传输成本。

可以从 S3 公有 DNS 域解析特定于端点的 S3 DNS 名称。

私有 DNS

VPC 接口端点的私有 DNS 选项简化了通过 VPC 端点路由 S3 流量的过程，并帮助您充分利用应用程序可用的最低成本网络路径。您可以使用私有 DNS 选项来路由区域 S3 流量，而无需更新 S3 客户端以使用接口端点的端点特定 DNS 名称，也无需管理 DNS 基础设施。启用私有 DNS 名称后，对于以下端点，区域 S3 DNS 查询将解析为 AWS PrivateLink 的私有 IP 地址：

- 区域存储桶端点（例如 `s3.us-east-1.amazonaws.com`）
- 控制端点（例如 `s3-control.us-east-1.amazonaws.com`）
- 接入点端点（例如 `s3-accesspoint.us-east-1.amazonaws.com`）

如果您的 VPC 中有网关端点，则可以通过现有 S3 网关端点自动路由 VPC 内请求，并通过接口端点自动路由本地请求。此方法允许您使用 VPC 内流量不计费的网关端点来优化网络成本。您的本地应用程序可以在入站解析器端点的帮助下使用 AWS PrivateLink。Amazon 为您的 VPC 提供 DNS 服务器，称为 Route 53 Resolver。入站解析器端点将来自本地网络的 DNS 查询转发到 Route 53 Resolver。

> **Important**
>
> 要在使用仅对入站端点启用私有 DNS 时充分利用成本最低的网络路径，您的 VPC 中必须存在网关端点。当选择仅对入站端点启用私有 DNS 选项时，网关端点的存在有助于确保 VPC 内流量始终通过 AWS 私有网络路由。当选择仅对入站端点启用私有 DNS 选项时，必须维护此网关端点。如果要删除网关端点，则必须先清除仅对入站端点启用私有 DNS。
> 如果您想要将现有接口端点更新为仅对入站端点启用私有 DNS，请先确认您的 VPC 具有 S3 网关端点。有关网关端点和管理私有 DNS 名称的更多信息，请分别参阅《AWS PrivateLink 指南》中的网关 VPC 端点和管理 DNS 名称。

仅对入站端点启用私有 DNS 选项仅适用于支持网关端点的服务。

有关创建使用仅对入站端点启用私有 DNS 的 VPC 端点的更多信息，请参阅《AWS PrivateLink 指南》中的创建接口端点。

使用 VPC 控制台

在控制台中，您有两个选项：启用 DNS 名称和仅对入站端点启用私有 DNS。启用 DNS 名称是 AWS PrivateLink 支持的选项。通过使用启用 DNS 名称选项，您可以使用 Amazon 的与 Amazon S3 的私有

连接，同时向默认公有端点 DNS 名称发出请求。启用此选项后，客户可以利用其应用程序可用的成本最低的网络路径。

当您在 Amazon S3 的现有或新的 VPC 接口端点上启用私有 DNS 名称时，默认情况下会选择仅对入站端点启用私有 DNS 选项。如果选择此选项，则您的应用程序仅对本地流量使用接口端点。这种 VPC 内流量会自动使用成本较低的网关端点。或者，您可以清除仅对入站端点启用私有 DNS，以通过您的接口端点路由所有 S3 请求。

使用 AWS CLI

如果不指定 `PrivateDnsOnlyForInboundResolverEndpoint` 的值，则它默认为 `true`。但是，在您的 VPC 应用设置之前，它会进行检查以确保 VPC 中存在网关端点。如果 VPC 中存在网关端点，则调用成功。否则，您将看到以下错误消息：

要将 PrivateDnsOnlyForInboundResolverEndpoint 设置为 true，VPC *vpce_id* 必须具有该服务的网关端点。

对于新的接口 VPC 端点

使用 `private-dns-enabled` 和 `dns-options` 属性通过命令行启用私有 DNS。`dns-options` 属性中的 `PrivateDnsOnlyForInboundResolverEndpoint` 选项必须设置为 `true`。将 *user input placeholders* 替换为您自己的信息。

```
aws ec2 create-vpc-endpoint \
--region us-east-1 \
--service-name s3-service-name \
--vpc-id client-vpc-id \
--subnet-ids client-subnet-id \
--vpc-endpoint-type Interface  \
--private-dns-enabled  \
--ip-address-type ip-address-type \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true \
--security-group-ids client-sg-id
```

对于现有 VPC 端点

如果您想对现有 VPC 端点使用私有 DNS，请使用以下示例命令并将 *user input placeholders* 替换为您自己的信息。

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=false
```

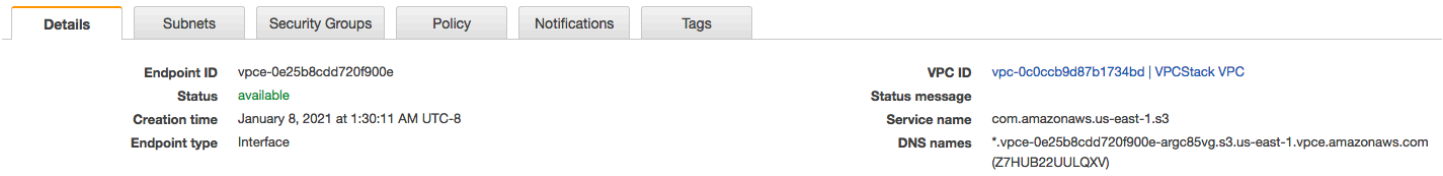
如果您想更新现有 VPC 端点以仅对入站解析器启用私有 DNS，请使用以下示例并将示例值替换为您自己的值。

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true
```

从 S3 接口端点访问存储桶、接入点和 Amazon S3 控制 API 操作

您可以使用 AWS CLI 或 AWS SDK 通过 S3 接口端点访问存储桶、S3 接入点和 Amazon S3 控制 API 操作。

下图显示了 VPC 控制台详细信息选项卡，您可以在其中找到 VPC 端点的 DNS 名称。
在此示例中，VPC 端点 ID (vpce-id) 为 `vpce-0e25b8cdd720f900e`，DNS 名称为 `*.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com`。



使用 DNS 名称访问资源时，将 * 替换为相应的值。用来代替 * 的相应值如下所示：

- `bucket`
- `accesspoint`
- `control`

例如，要访问存储桶，请使用如下所示的 DNS 名称。

```
bucket.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com
```

有关如何使用 DNS 名称访问存储桶、接入点和 Amazon S3 控制 API 操作的示例，请参阅下面的章节：AWS CLI 示例和AWS SDK 示例。

有关如何查看端点特定 DNS 名称的更多信息，请参阅《VPC 用户指南》中的查看端点服务私有 DNS 名称配置。

AWS CLI 示例

要在 AWS CLI 命令中通过 S3 接口端点访问 S3 存储桶、S3 接入点或 Amazon S3 控制 API操作，请使用 `--region` 和 `--endpoint-url` 参数。

示例：使用端点 URL 列出存储桶中的对象

在以下示例中，将存储桶名称 *my-bucket*、区域 *us-east-1* 和 VPC 端点 ID 的 DNS 名称 *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 替换为您自己的信息。

```
aws s3 ls s3://my-bucket/ --region us-east-1 --endpoint-url
 https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
```

示例：使用端点 URL 列出接入点中的对象

- 方法 1 – 将接入点的 Amazon 资源名称（ARN）与接入点端点结合使用

  将 ARN *us-east-1:123456789012:accesspoint/accesspointexamplename*、区域 *us-east-1* 和 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 替换为您自己的信息。

  ```
  aws s3api list-objects-v2 --bucket arn:aws:s3:us-east-1:123456789012:accesspoint/
  accesspointexamplename --region us-east-1 --endpoint-url
   https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
  ```

  如果您无法成功运行该命令，请将 AWS CLI 更新为最新版本，然后重试。有关更新说明的更多信息，请参阅《AWS Command Line Interface 用户指南》中的安装或更新 AWS CLI 的最新版本。
- 方法 2 – 将接入点的别名与区域存储桶端点结合使用

  在以下示例中，将接入点别名 *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*、区域 *us-east-1* 和 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 替换为您自己的信息。

```
aws s3api list-objects-v2 --
bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias
 --region us-east-1 --endpoint-url https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com
```

- 方法 3 – 将接入点的别名与接入点端点结合使用

  首先，要构造一个包含存储桶作为主机名一部分的 S3 端点，请将寻址样式设置为 `virtual` 以供 `aws s3api` 使用。有关 `AWS configure` 的更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置和凭证文件设置。

  ```
  aws configure set default.s3.addressing_style virtual
  ```

  然后，在以下示例中，将接入点别名 *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*、区域 *us-east-1* 和 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 替换为您自己的信息。有关接入点别名的更多信息，请参阅为您的 S3 存储桶接入点使用存储桶式别名

  ```
  aws s3api list-objects-v2 --
  bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias --
  region us-east-1 --endpoint-url https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
  east-1.vpce.amazonaws.com
  ```

示例：使用端点 URL 通过 S3 控制 API 操作列出任务

在以下示例中，将区域 *us-east-1*、VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和账户 ID *12345678* 替换为您自己的信息。

```
aws s3control --region us-east-1 --endpoint-url
 https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com list-jobs --
account-id 12345678
```

AWS SDK 示例

要在使用 AWS SDK 时通过 S3 接口端点访问 S3 存储桶、S3 接入点、Amazon S3 控制 API 操作，请将 SDK 更新到最新版本。然后，将客户端配置为使用端点 URL 通过 S3 接口端点访问存储桶、接入点或 Amazon S3 控制 API 操作。

SDK for Python (Boto3)

示例：使用端点 URL 访问 S3 存储桶

在以下示例中，将区域 *us-east-1* 和 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 替换为您自己的信息。

```
s3_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

示例：使用端点 URL 访问 S3 接入点

在以下示例中，将区域 *us-east-1* 和 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 替换为您自己的信息。

```
ap_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com'
)
```

示例：使用端点 URL 访问 Amazon S3 控制 API

在以下示例中，将区域 *us-east-1* 和 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 替换为您自己的信息。

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

SDK for Java 1.x

示例：使用端点 URL 访问 S3 存储桶

在以下示例中，将 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 替换为您自己的信息。

```
// bucket client
final AmazonS3 s3 = AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
List<Bucket> buckets = s3.listBuckets();
```

示例：使用端点 URL 访问 S3 接入点

在以下示例中，将 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和 ARN *us-east-1:123456789012:accesspoint/prod* 替换为您自己的信息。

```
// accesspoint client
final AmazonS3 s3accesspoint =
 AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
ObjectListing objects = s3accesspoint.listObjects("arn:aws:s3:us-east-1:123456789012:accesspoint/prod");
```

示例：使用端点 URL 访问 Amazon S3 控制 API 操作

在以下示例中，将 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 替换为您自己的信息。

```
// control client
final AWSS3Control s3control =
 AWSS3ControlClient.builder().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
```

```
final ListJobsResult jobs = s3control.listJobs(new ListJobsRequest());
```

SDK for Java 2.x

示例：使用端点 URL 访问 S3 存储桶

在以下示例中，将 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和区域 *Region.US_EAST_1* 替换为您自己的信息。

```
// bucket client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com"))
                    .build()
```

示例：使用端点 URL 访问 S3 接入点

在以下示例中，将 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和区域 *Region.US_EAST_1* 替换为您自己的信息。

```
// accesspoint client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com"))
                    .build()
```

示例：使用端点 URL 访问 Amazon S3 控制 API

在以下示例中，将 VPC 端点 ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和区域 *Region.US_EAST_1* 替换为您自己的信息。

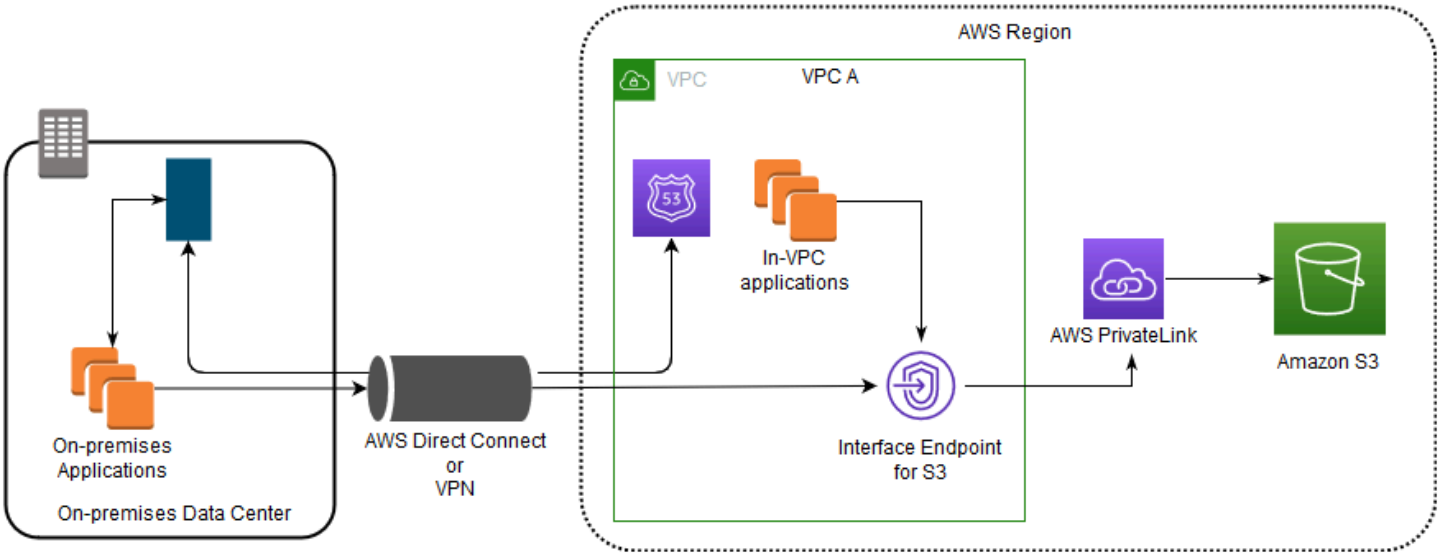
```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com"))
                                    .build()
```

更新本地 DNS 配置

使用特定于端点的 DNS 名称访问适用于 Amazon S3 的接口端点时，您无需更新本地 DNS 解析程序。您可以使用来自公有 Amazon S3 DNS 域的接口端点的私有 IP 地址解析特定于端点的 DNS 名称。

使用接口端点访问 Amazon S3，无需 VPC 中的网关端点和互联网网关

VPC 中的接口端点可以通过 Amazon 网络将 VPC 内的应用程序和本地应用程序路由到 Amazon S3，如下图所示。

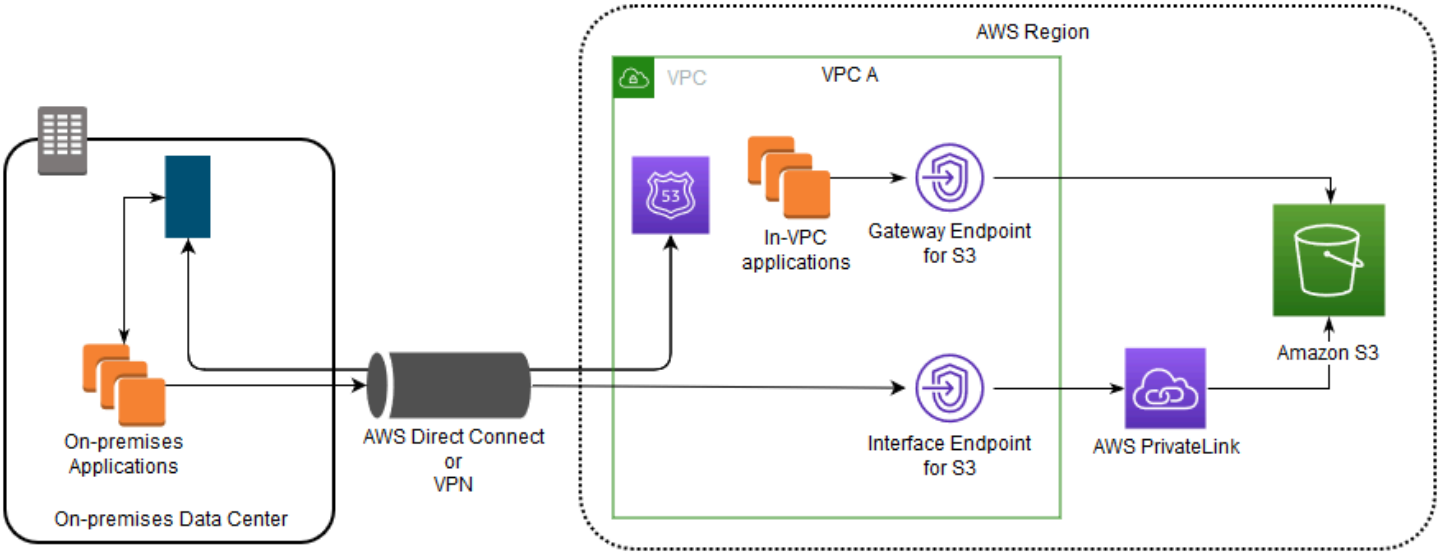


该图阐释了以下内容：

- 您的本地部署网络使用 AWS Direct Connect 或者 AWS VPN 连接到 VPC A。
- 本地和 VPC A 中的应用程序使用特定于端点的 DNS 名称通过 S3 接口端点访问 Amazon S3。
- 本地部署应用程序通过 AWS Direct Connect（或 AWS VPN）将数据发送到 VPC 中的接口端点。AWS PrivateLink 通过 AWS 网络将数据从接口端点移动到 Amazon S3。
- VPC 中的应用程序还向接口端点发送通信。AWS PrivateLink 通过 AWS 网络将数据从接口端点移动到 Amazon S3。

在同一 VPC 中同时使用网关端点和接口端点来访问 Amazon S3

您可以创建接口端点并将现有网关端点保留在同一 VPC 中，如下图所示。通过这种方法，您可以允许 VPC 内应用程序继续通过网关端点访问 Amazon S3，而无需付费。然后，只有您的本地应用程序才会使用接口端点访问 Amazon S3。要通过这种方式访问 Amazon S3，您必须更新本地应用程序，以使用适用于 Amazon S3 的特定于端点的 DNS 名称。



该图阐释了以下内容：

- 本地部署应用程序使用特定于端点的 DNS 名称通过 AWS Direct Connect（或 AWS VPN）将数据发送到 VPC 中的接口端点。AWS PrivateLink 通过 AWS 网络将数据从接口端点移动到 Amazon S3。
- 使用默认的区域 Amazon S3 名称，VPC 内应用程序会将数据发送到通过 AWS 网络连接到 Amazon S3 的网关端点。

有关网关端点的更多信息，请参阅 VPC 用户指南中的网关 VPC 端点。

为 Amazon S3 创建 VPC 端点策略

您可以为 VPC 端点附加控制对 Amazon S3 的访问的端点策略。该策略指定以下信息：

- 可执行操作的 AWS Identity and Access Management (IAM) 主体
- 可执行的操作
- 可对其执行操作的资源

您还可以使用 Amazon S3 存储桶策略，通过使用存储桶策略中的 `aws:sourceVpce` 条件限制从特定 VPC 端点访问特定存储桶。以下示例显示了限制对存储桶或端点的访问的策略。

主题

- 示例：限制从 VPC 端点对特定存储桶的访问
- 示例：限制从 VPC 端点对特定账户中存储桶的访问
- 示例：限制对 S3 存储桶策略中特定 VPC 端点的访问

示例：限制从 VPC 端点对特定存储桶的访问

您可以创建一个端点策略来仅允许访问特定的 Amazon S3 存储桶。如果您的 VPC 中有使用存储桶的其他 AWS 服务，这种策略会非常有用。以下存储桶策略限制为仅可访问 *amzn-s3-demo-bucket1*。要使用此端点策略，请将 *amzn-s3-demo-bucket1* 替换为您的存储桶名称。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket1",
                   "arn:aws:s3:::amzn-s3-demo-bucket1/*"]
    }
  ]
}
```

示例：限制从 VPC 端点对特定账户中存储桶的访问

您可以创建一个端点策略，以限制仅访问特定 AWS 账户中的 S3 存储桶。要防止 VPC 内的客户端访问并非您拥有的存储桶，请在端点策略中使用以下语句。以下示例语句创建了一个策略，此策略将访问限制为由单个 AWS 账户 ID *111122223333* 拥有的资源。

```
{
  "Statement": [
    {
      "Sid": "Access-to-bucket-in-specific-account-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:s3:::*",
      "Condition": {
        "StringNotEquals": {
```

```
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

> **Note**
>
> 要指定要访问的资源的 AWS 账户 ID，可以在 IAM 策略中使用 `aws:ResourceAccount` 或 `s3:ResourceAccount` 键。但请注意，有些 AWS 服务依赖于访问 AWS 托管式存储桶。因此，在 IAM 策略中使用 `aws:ResourceAccount` 或 `s3:ResourceAccount` 键也可能会影响对这些资源的访问。

示例：限制对 S3 存储桶策略中特定 VPC 端点的访问

示例：限制对 S3 存储桶策略中特定 VPC 端点的访问

以下 Amazon S3 存储桶策略仅允许从 VPC 端点 *vpce-1a2b3c4d* 访问特定的存储桶 *amzn-s3-demo-bucket2*。如果未使用指定的端点，则该策略拒绝对存储桶的所有访问。`aws:sourceVpce` 条件指定端点，而不需要 VPC 端点资源的 Amazon 资源名称（ARN），只需要端点 ID。要使用此存储桶策略，请将 *amzn-s3-demo-bucket2* 和 *vpce-1a2b3c4d* 替换为您的存储桶名称和端点。

> **Important**
>
> - 当应用以下 Amazon S3 存储桶策略来限制仅可访问特定的 VPC 端点时，您可能会无意中屏蔽对存储桶的访问。存储桶策略旨在专门限制存储桶访问源自 VPC 端点的连接，而这可能会屏蔽到存储桶的所有连接。有关如何修复此问题的信息，请参阅我的存储桶策略有错误的 VPC 或 VPC 端点 ID。AWS Support 知识中心内的如何修复策略才能访问存储桶？。
> - 在使用以下示例策略之前，将 VPC 端点 ID 替换为适合您的使用案例的值。否则，您将无法访问您的存储桶。
> - 此策略禁用控制台访问指定的存储桶，因为控制台请求不是来自指定的 VPC 端点。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909152",
```

```
  "Statement": [
    { "Sid": "Access-to-specific-VPCE-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket2",
                   "arn:aws:s3:::amzn-s3-demo-bucket2/*"],
      "Condition": {"StringNotEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}}
    }
  ]
}
```

有关更多策略示例，请参阅 VPC 用户指南中的 Amazon S3 端点。

有关 VPC 连接的更多信息，请参阅 AWS 白皮书 Amazon Virtual Private Cloud 连接性选项中的 Network-to-VPC connectivity options（从网络到 VPC 的连接性选项）。

访问控制

在 AWS 中，资源是您可使用的实体。在 Amazon Simple Storage Service（S3）中，存储桶和对象是最初的 Amazon S3 资源。每个 S3 客户可能都有包含对象的存储桶。当有新功能添加到 S3 时，则会添加额外的资源，但并非每个客户都使用这些特定于功能的资源。有关 Amazon S3 资源的更多信息，请参阅 S3 资源。

默认情况下，所有 Amazon S3 资源都是私有的。默认情况下，创建资源的 AWS 账户的根用户（资源拥有者）和该账户中具有必要权限的 IAM 用户可以访问他们创建的资源。资源拥有者决定还有谁可以访问该资源以及允许其他人对该资源执行哪些操作。S3 有各种访问管理工具，您可以使用这些工具向其他人授予对您的 S3 资源的访问权限。

以下各节概述了 S3 资源、可用的 S3 访问管理工具以及每种访问管理工具的最佳使用案例。这些章节中的列表力求全面，旨在包括所有 S3 资源、访问管理工具和常见的访问管理使用案例。同时，这些章节设计成目录，可引导您找到所需的技术细节。如果您对以下某些主题有很好的理解，则可以跳到适用于您的章节。

主题

- S3 资源
- 身份
- 访问管理工具
- 操作

- 访问管理使用案例
- 访问管理故障排除

S3 资源

原始 Amazon S3 资源是存储桶及其中包含的对象。随着向 S3 添加新功能，也会添加新的资源。以下是 S3 资源及相应功能的完整列表。

资源类型	Amazon S3 功能	描述
bucket	核心功能	存储桶是对象的容器。要将对象存储在 S3 中，请创建存储桶，然后将一个或多个对象上传到该存储桶。有关更多信息，请参阅 创建、配置和使用 Amazon S3 存储桶。
object		对象可以是一个文件和描述该文件的任何元数据。当对象位于存储桶中时，您可以将其打开、下载并移动它。有关更多信息，请参阅 上传、下载和处理 Amazon S3 中的对象。
accesspoint	访问点	接入点是附加到存储桶的命名网络端点，您可以使用这些存储桶执行 Amazon S3 object 操作（如 GetObject 和 PutObject ）。每个接入点都有不同的权限、网络控制和自定义的接入点策略，该策略与附加到底层存储桶的存储桶策略结合使用。您可以将任何接入点配置为仅接受来自虚拟私有云（VPC）的请求，或者为每个接入点配置自定义屏蔽公共访问权限设置。有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。
objectlambdaaccesspoint		对象 Lambda 接入点是也与 Lambda 函数相关联的存储桶的接入点。借助对象 Lambda 接入点，您可以将自己的代码添加到 Amazon S3 GET、LIST 和 HEAD 请求中，以便在数据返回到应用程序时修改和处理数据。有关更多信息，请参阅 创建对象 Lambda 接入点。
multiregionaccesspoint		多区域接入点提供了一个全局端点，应用程序可以使用该端点来满足来自位于多个 AWS 区域的 Amazon S3 存储桶的请求。您可以使用多区域接入点通过单个区域中使用的相同架

资源类型	Amazon S3 功能	描述
		构构建多区域应用程序，然后在世界任何地方运行这些应用程序。向多区域接入点全局端点发出的应用程序请求自动通过 AWS 全局网络路由到距离最近的 Amazon S3 存储桶，而不是通过拥挤的公共互联网发送请求。有关更多信息，请参阅 Amazon S3 中的多区域接入点。
job	S3 批量操作	任务是 S3 批量操作功能的资源。您可以使用 S3 批量操作对您指定的 Amazon S3 对象列表执行大规模批量操作。Amazon S3 跟踪批量操作任务的进度，发送通知，并存储所有操作的详细完成报告，从而提供完全托管式、可审计的无服务器体验。有关更多信息，请参阅 对 Amazon S3 对象执行大规模批量操作。
storagelensconfiguration	S3 Storage Lens 存储统计管理工具	S3 Storage Lens 存储分析功能配置负责收集组织范围的存储指标和跨账户的用户数据。S3 Storage Lens 存储分析功能为管理员提供了组织中数百甚至数千个账户的对象存储使用情况和活动的单一视图，并提供了可在多个聚合级别生成见解的详细信息。有关更多信息，请参阅 使用 Amazon S3 Storage Lens 存储统计管理工具评估您的存储活动和使用情况。
storagelensgroup		S3 Storage Lens 组使用基于对象元数据的自定义筛选条件来聚合指标。S3 Storage Lens 组可协助您调查数据的特征，例如按龄期划分的对象分布、最常见的文件类型等。有关更多信息，请参阅 使用 S3 Storage Lens 组。
accessgrantsinstance	S3 访问授权	S3 Access Grants 实例是您创建的 S3 授权的容器。借助 S3 Access Grants，您可以为您账户中的 IAM 身份、其它账户（跨账户）中的 IAM 身份以及从公司目录中添加到 AWS IAM Identity Center 的目录身份创建对于您的 Amazon S3 数据的授权。有关 S3 Access Grants 的更多信息，请参阅使用 S3 Access Grants 管理访问权限。

资源类型	Amazon S3 功能	描述
accessgrantslocation		Access Grants 位置是存储桶、存储桶内的前缀或您在 S3 Access Grants 实例中注册的对象。您必须在 S3 Access Grants 实例中注册位置，然后才能创建对该位置的授权。然后，借助 S3 Access Grants，您可以为您账户中的 IAM 身份、其它账户（跨账户）中的 IAM 身份以及从公司目录中添加到 AWS IAM Identity Center 的目录身份授予对存储桶、前缀或对象的访问权限。有关 S3 Access Grants 的更多信息，请参阅使用 S3 Access Grants 管理访问权限。
accessgrant		访问授权是对您的 Amazon S3 数据的单独授权。借助 S3 Access Grants，您可以为您账户中的 IAM 身份、其它账户（跨账户）中的 IAM 身份以及从公司目录中添加到 AWS IAM Identity Center 的目录身份创建对于您的 Amazon S3 数据的授权。有关 S3 Access Grants 的更多信息，请参阅使用 S3 Access Grants 管理访问权限。

存储桶

Amazon S3 存储桶有两种类型：通用型存储桶 和目录存储桶。

- 通用存储桶是最初的 S3 存储桶类型，建议用于大多数使用案例和访问模式。通用存储桶还允许跨所有存储类（S3 Express One Zone 除外）存储对象。有关 S3 存储类的更多信息，请参阅使用 Amazon S3 存储类。
- 目录存储桶使用 S3 Express 单区存储类，如果您的应用程序注重性能，并且受益于个位数毫秒的 PUT 和 GET 延迟，则建议使用该存储类。有关更多信息，请参阅 目录桶、什么是 S3 Express One Zone？ 和 适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

对 S3 资源进行分类

Amazon S3 提供了对您的 S3 资源进行分类和整理的功能。对资源进行分类不仅对整理资源很有用，还可以根据资源类别设置访问管理规则。特别是，前缀和标记是设置访问管理权限时可以使用的两个存储整理功能。

> **Note**
>
> 以下信息适用于通用型存储桶。目录存储桶不支持标记，并且有前缀限制。有关更多信息，请参阅 适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

- 前缀 – Amazon S3 中的前缀是对象键名称开头的一个字符串，用于整理存储在 S3 存储桶中的对象。您可以使用分隔符 [例如正斜杠（/）] 来表示对象键名称中前缀的结尾。例如，您可能有以 `engineering/` 前缀开头的对象键名称或以 `marketing/campaigns/` 前缀开头的对象键名称。在前缀末尾使用分隔符（例如正斜杠字符 /）可以模拟文件夹和文件命名约定。但是，在 S3 中，前缀是对象键名称的一部分。在通用型 S3 存储桶中，没有实际的文件夹层次结构。

  Amazon S3 支持使用对象的前缀对于对象进行整理和分组。您还可以通过对象的前缀管理对于对象的访问权限。例如，您可以将访问权限限制为仅限名称以特定前缀开头的对象。

  有关更多信息，请参阅 使用前缀组织对象。S3 控制台使用文件夹 的概念，在通用型存储桶中，文件夹本质上是附加在对象键名称之前的前缀。有关更多信息，请参阅 使用文件夹在 Amazon S3 控制台中整理对象。
- 标签 – 每个标签都是您分配给资源的键值对。例如，您可以使用标签 `topicCategory=engineering` 标记某些资源。您可以使用标记来协助进行成本分配、分类和整理以及访问控制。存储桶标记仅用于成本分配。您可以标记对象、Storage Lens 存储分析功能、任务和 S3 Access Grants，以便进行整理或实施访问控制。在 S3 Access Grants 中，您还可以使用标记进行成本分配。举个使用标签控制资源访问权限的示例，您只能共享具有特定标签或标签组合的对象。

  有关更多信息，请参阅《IAM 用户指南》中的使用资源标签控制对 AWS 资源的访问。

身份

在 Amazon S3 中，资源拥有者是创建资源（例如存储桶或对象）的身份。默认情况下，只有创建资源的账户的根用户和该账户中具有所需权限的 IAM 身份才能访问 S3 资源。资源拥有者可以向其它身份授予对其 S3 资源的访问权限。

不拥有资源的身份可以请求访问该资源。对资源的请求已经过身份验证或未经过身份验证。经过身份验证的请求必须包含可验证请求发送者身份的签名值，但未经身份验证的请求不需要签名。我们建议您仅向经过身份验证的用户授予访问权限。有关请求身份验证的更多信息，请参阅 提出请求。

> **Important**
>
> 我们建议不要使用 AWS 账户根用户凭证发起经过身份验证的请求。而应创建一个 IAM 角色并授予该角色完全访问权限。我们将具有此角色的用户称为管理员用户。您可以使用分配给管理员角色的用户凭证而不是 AWS 账户 根用户凭证来与 AWS 交互和执行任务，例如创建存储桶、创建用户以及授予权限。有关更多信息，请参阅《AWS 一般参考》中的 AWS 账户 根用户凭证和 IAM 用户凭证，并参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

在 Amazon S3 中访问您的数据的身份可以是以下身份之一：

AWS 账户拥有者

创建资源的 AWS 账户。例如，创建存储桶的账户。此账户是该资源的拥有者。有关更多信息，请参阅 AWS 账户根用户。

与 AWS 账户拥有者同属一个账户的 IAM 身份

在为需要 S3 访问权限的新团队成员设置账户时，AWS 账户拥有者可以使用 AWS Identity and Access Management（IAM）来创建用户、组和角色。然后，AWS 账户拥有者可以与这些 IAM 身份共享资源。账户拥有者还可以指定要向 IAM 身份授予的权限，这些权限允许或拒绝可以对共享资源执行的操作。

IAM 身份提供了增强的功能，包括要求用户在访问共享资源之前先输入登录凭证的功能。通过使用 IAM 身份，您可以实施一种形式的 IAM 多重身份验证（MFA）来支持强大的身份基础。IAM 最佳实践是创建用于访问管理的角色，而不是向每个单独用户授予权限。您可以将各个用户分配给相应的角色。有关更多信息，请参阅 IAM 安全最佳实践。

其它 AWS 账户拥有者及其 IAM 身份（跨账户存取）

AWS 账户拥有者还可以向其它 AWS 账户拥有者或属于其它 AWS 账户的 IAM 身份授予对资源的访问权限。

> **Note**
>
> 权限委托 – 如果某个 AWS 账户拥有一项资源，则该账户可以将这些权限授予其它 AWS 账户。然后，这个账户就可以将这些权限或其子集委托给同一账户中的用户。这称为 权限委托。但从另一账户接收权限的账户不能“跨账户”向其它 AWS 账户委托这些权限。

匿名用户（公共访问权限）

AWS 账户拥有者可以公开资源。从技术上讲，公开某资源会与匿名用户 共享该资源。除非您更改此设置，否则默认情况下，自 2023 年 4 月以来创建的存储桶会屏蔽所有公共访问权限。我们建议您将存储桶设置为屏蔽公共访问权限，并且仅向经过身份验证的用户授予访问权限。有关屏蔽公共访问权限的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

AWS 服务

资源拥有者可以向其它 AWS 服务授予对 Amazon S3 资源的访问权限。例如，您可以向 AWS CloudTrail 服务授予将日志文件写入存储桶的 `s3:PutObject` 权限。有关更多信息，请参阅向 AWS 服务提供访问权限。

公司目录身份

资源拥有者可以使用 S3 Access Grants 向公司目录中的用户或角色授予对 S3 资源的访问权限。有关将公司目录添加到 AWS IAM Identity Center 的更多信息，请参阅 What is IAM Identity Center?

存储桶拥有者或资源拥有者

用于创建存储桶和上传对象的 AWS 账户拥有这些资源。存储桶拥有者可以向其他 AWS 账户（或其他账户中的用户）授予上传对象的跨账户权限。

当存储桶拥有者允许其它账户向存储桶上传对象时，默认情况下，该存储桶拥有者拥有上传到其存储桶的所有对象。但是，如果强制存储桶拥有者 和存储桶拥有者优先 存储桶设置都处于关闭状态，则上传对象的 AWS 账户将拥有这些对象，而存储桶拥有者对其它账户拥有的对象没有权限，但以下情况除外：

- 账单由存储桶拥有者支付。存储桶拥有者可以拒绝对任何对象的访问，或删除存储桶中的任何对象，而无论它们的拥有者是谁。
- 存储桶拥有者可以归档任何对象或还原归档的对象，而无论它们的拥有者是谁。归档是指用于存储对象的存储类。有关更多信息，请参阅 管理存储生命周期。

访问管理工具

Amazon S3 提供了各种安全功能和工具。以下是这些功能和工具的完整列表。您并不需要所有这些访问管理工具，但必须使用一个或多个工具来授予对您的 Amazon S3 资源的访问权限。正确应用这些工具有助于确保只有目标用户才能访问您的资源。

最常用的访问管理工具是访问策略。访问策略可以是附加到 AWS 资源的基于资源的策略，例如存储桶的存储桶策略。访问策略也可以是附加到 AWS Identity and Access Management（IAM）身份（如 IAM 用户、组或角色）的基于身份的策略。编写访问策略来向 AWS 账户和 IAM 用户、组和角色授予对资源执行操作的权限。例如，您可以向另一个 AWS 账户授予 PUT Object 权限，以使该账户可以向您的存储桶上传对象。

访问策略描述了谁可以访问哪些内容。当 Amazon S3 收到请求时，它必须评估所有访问策略，来决定是授权还是拒绝该请求。有关 Amazon S3 如何评估这些策略的更多信息，请参阅 Amazon S3 如何对请求授权。

以下是 Amazon S3 中提供的访问管理工具。

存储桶策略

Amazon S3 存储桶策略是采用 JSON 格式的 AWS Identity and Access Management（IAM）基于资源的策略，该策略附加到特定的存储桶。使用存储桶策略来向其它 AWS 账户或 IAM 身份授予对相应存储桶及其中对象的权限。使用存储桶策略可以满足许多 S3 访问管理使用案例的要求。通过存储桶策略，您可以对存储桶访问权限进行个性化设置，以协助确保只有您已批准的身份才能访问资源并在其中执行操作。有关更多信息，请参阅 Amazon S3 的存储桶策略。

下面是一个示例存储桶策略。您可以使用 JSON 文件来表示存储桶策略。此示例策略向 IAM 角色授予对存储桶中所有对象的读取权限。该策略包含一条名为 BucketLevelReadPermissions 的语句，该语句允许对名为 amzn-s3-demo-bucket1 的存储桶中的对象执行 s3:GetObject 操作（读取权限）。通过将 IAM 角色指定为 Principal，此策略向具有该角色的任何 IAM 用户授予访问权限。要使用此示例策略，请将 *user input placeholders* 替换为您自己的信息。

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"BucketLevelReadPermissions",
      "Effect":"Allow",
      "Principal": {
     "AWS": "arn:aws:iam::123456789101:role/s3-role"
      },
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::amzn-s3-demo-bucket1/*"]
    }]
  }
```

> **Note**
>
> 创建策略时，请避免在 `Principal` 元素中使用通配符（`*`），因为使用通配符将允许任何人访问您的 Amazon S3 资源。而是应显式列出允许访问存储桶的用户或组，或者使用策略中的条件子句列出必须满足的条件。此外，在适用的情况下，向用户或组授予特定的权限，而不是包含通配符来让用户或组执行操作。

基于身份的策略

基于身份的策略或 IAM 用户策略是一种 AWS Identity and Access Management（IAM）策略。基于身份的策略是采用 JSON 格式的策略，该策略附加到您的 AWS 账户中的 IAM 用户、组或角色。您可以使用基于身份的策略，来向 IAM 身份授予对您的存储桶或对象的访问权限。您可以在您的账户中创建 IAM 用户、组和角色，并为其附加访问策略。然后，您可以授予对 AWS 资源（包括 Amazon S3 资源）的访问权限。有关更多信息，请参阅 Amazon S3 基于身份的策略。

下面是基于身份的策略的示例。此示例策略允许关联的 IAM 角色对存储桶及其中的对象执行六种不同的 Amazon S3 操作（权限）。如果您将此策略附加到您账户中的 IAM 角色，并将该角色分配给您的某些 IAM 用户，则具有此角色的用户将能够对在策略中指定的资源（存储桶）执行这些操作。要使用此示例策略，请将 *`user input placeholders`* 替换为您自己的信息。

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Sid": "AssignARoleActions",
    "Effect": "Allow",
    "Action": [
   "s3:PutObject",
   "s3:GetObject",
   "s3:ListBucket",
   "s3:DeleteObject",
   "s3:GetBucketLocation"
    ],
    "Resource": [
   "arn:aws:s3:::amzn-s3-demo-bucket1/*",
   "arn:aws:s3:::amzn-s3-demo-bucket1"
    ]
    },
  {
    "Sid": "AssignARoleActions2",
```

```
    "Effect": "Allow",
    "Action": "s3:ListAllMyBuckets",
    "Resource": "*"
  }
  ]
}
```

S3 访问授权

使用 S3 Access Grants 为企业身份目录（例如 Active Directory）中的身份以及 AWS Identity and Access Management（IAM）身份创建对您的 Amazon S3 数据的访问授权。S3 Access Grants 可协助您大规模管理数据权限。此外，S3 Access Grants 还会记录最终用户身份以及用于访问 AWS CloudTrail 中 S3 数据的应用程序。这提供了详细的审计历史记录，细至与 S3 存储桶中数据的所有访问权限对应的最终用户身份。有关更多信息，请参阅 使用 S3 Access Grants 管理访问权限。

访问点

对于使用 S3 上的共享数据集的应用程序，Amazon S3 接入点可简化大规模管理数据访问的事宜。接入点是附加到存储桶的命名网络端点。您可以使用接入点来大规模执行 S3 对象操作，例如上传和检索对象。一个存储桶最多可附加 10000 个接入点，对于每个接入点，您可以强制实施不同的权限和网络控制，从而让您可以详细控制对于 S3 对象的访问权限。S3 接入点可以与同一个账户或其它可信账户中的存储桶相关联。接入点策略是与底层存储桶策略一起进行评估的基于资源的策略。有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。

访问控制列表 (ACL)

ACL 是用于确定被授权者和所授予权限的授权列表。ACL 向其它 AWS 账户授予基本的读取或写入权限。ACL 使用特定于 Amazon S3 的 XML 架构。ACL 是一种 AWS Identity and Access Management（IAM）策略。对象 ACL 用于管理对于对象的访问权限，而存储桶 ACL 用于管理对存储桶的访问权限。使用存储桶策略，整个存储桶只有单个策略，但可以为每个对象指定对象 ACL。我们建议您将 ACL 保持为关闭状态，除非有必须单独控制每个对象的访问权限的特殊情况。有关 ACL 的更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

> **Warning**
>
> Amazon S3 中的大多数现代使用案例不需要使用 ACL。

下面是一个示例存储桶 ACL。ACL 中的授权显示一个具有完全控制权限的存储桶拥有者。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
 <AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
   <ID>Owner-Canonical-User-ID</ID>
   <DisplayName>owner-display-name</DisplayName>
  </Owner>
 <AccessControlList>
  <Grant>
   <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Canonical
 User">
    <ID>Owner-Canonical-User-ID</ID>
    <DisplayName>display-name</DisplayName>
   </Grantee>
   <Permission>FULL_CONTROL</Permission>
  </Grant>
 </AccessControlList>
</AccessControlPolicy>
```

对象所有权

要管理对于对象的访问权限，您必须是该对象的拥有者。您可以使用对象所有权存储桶级别设置，来控制上传到存储桶的对象的所有权。此外，使用对象所有权来开启 ACL。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于关闭状态。关闭 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并独占管理对数据的访问权限。为了管理访问权限，存储桶拥有者使用策略或其它访问管理工具（ACL 除外）。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

对象所有权有三个设置，您可以使用它们来控制上传到存储桶的对象的所有权并开启 ACL：

ACL 已关闭

- 强制存储桶拥有者（默认）– ACL 已关闭，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不影响对 S3 存储桶中数据的权限。存储桶专门使用策略来定义访问控制。

ACL 已开启

- Bucket owner preferred（首选存储桶拥有者）— 存储桶拥有者拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 写入存储桶的新对象。
- 对象编写者— 该 AWS 账户上传对象拥有该对象，对其拥有完全控制权，并且可以通过 ACL 授予其他用户访问该对象的权限。

其他最佳实践

考虑使用以下存储桶设置和工具来协助保护传输中数据和静态数据，这两者对于保持数据的完整性和可访问性至关重要：

- 屏蔽公共访问权限 - 请勿关闭默认的存储桶级别设置屏蔽公共访问权限。默认情况下，此设置会屏蔽对您的数据的公共访问权限。有关屏蔽公共访问权限的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。
- S3 版本控制 – 为确保数据完整性，您可以实现 S3 版本控制存储桶设置，此设置在您进行更新时对于对象进行版本控制，而不是覆盖它们。如果需要，您可以使用 S3 版本控制功能来保留、检索和还原以前的版本。有关 S3 版本控制的信息，请参阅在 S3 存储桶中使用版本控制。
- S3 对象锁定 – S3 对象锁定是您可以实施来实现数据完整性的另一个设置。此功能可以实现一次写入多次读取（WORM）模式，从而以不可变的方式存储对象。有关对象锁定的信息，请参阅使用 S3 对象锁定。
- 对象加密 – Amazon S3 提供了多种对象加密选项，用于保护传输中数据和静态数据。服务器端加密会在将对象保存到数据中心的磁盘上之前为对象加密，然后在下载对象时对数据进行解密。如果您验证了您的请求并且您拥有访问权限，则您访问加密对象或未加密对象的方式就没有区别。有关更多信息，请参阅 使用服务器端加密保护数据。默认情况下，S3 会加密新上传的对象。有关更多信息，请参阅 为 Amazon S3 存储桶设置默认服务器端加密行为。客户端加密是在将数据发送到 Amazon S3 之前加密数据的行为。有关更多信息，请参阅 使用客户端加密保护数据。
- 签名方法 – 签名版本 4 是将身份验证信息添加到通过 HTTP 发送的 AWS 请求的过程。出于安全考虑，大多数 AWS 请求都必须使用访问密钥（包括访问密钥 ID 和秘密访问密钥）进行签名。这两个密钥通常称为您的安全凭证。有关更多信息，请参阅对请求进行身份验证（AWS Signature Version 4）和 Signature Version 4 签名过程。

操作

有关 S3 权限和条件键的完整列表，请参阅《服务授权参考》中的 Actions, resources, and condition keys for Amazon S3。

操作

Amazon S3 的 AWS Identity and Access Management（IAM）操作是可以对 S3 存储桶或对象执行的可能操作。您可以向身份授予这些操作，以便这些身份可以对您的 S3 资源执行操作。S3 操作的示例为 `s3:GetObject`（读取存储桶中的对象）和 `s3:PutObject`（将对象写入存储桶）。

条件键

除操作外，IAM 条件键限制为仅在满足条件时才授予访问权限。条件键是可选的。

> **Note**
>
> 在基于资源的访问策略（例如存储桶策略）或基于身份的策略中，您可以指定以下内容：
>
> - 策略语句的 `Action` 元素中的一个操作或一系列操作。
> - 在策略语句的 `Effect` 元素中，您可以指定 `Allow` 来授予列出的操作，也可以指定 `Deny` 来阻止列出的操作。为进一步保持最低权限的做法，访问策略的 `Effect` 元素中的 `Deny` 语句应尽可能宽泛，而 `Allow` 语句应尽可能受限。与 `s3:*` 操作配对的 `Deny` 效果是为策略条件语句中包含的身份实现选择加入最佳实践的另一个好方法。
> - 策略语句的 `Condition` 元素中的条件键。

访问管理使用案例

Amazon S3 为资源拥有者提供了各种用于授予访问权限的工具。您使用的 S3 访问管理工具取决于您要共享的 S3 资源、您要授予访问权限的身份以及您想要允许或拒绝的操作。您可能希望使用一种 S3 访问管理工具或组合使用多种 S3 访问管理工具，来管理对 S3 资源的访问权限。

大多数情况下，您可以使用访问策略来管理权限。访问策略可以是基于资源的策略，此策略附加到资源（例如存储桶）或其它 Amazon S3 资源（S3 资源）。访问策略也可以是基于身份的策略，该策略附加到您账户中的 AWS Identity and Access Management（IAM）用户、组或角色。您可能会发现存储桶策略更适合您的使用案例。有关更多信息，请参阅 Amazon S3 的存储桶策略。或者，使用 AWS Identity and Access Management（IAM），您可以在您的 AWS 账户中创建 IAM 用户、组和角色，并通过基于身份的策略管理它们对存储桶和对象的访问权限。有关更多信息，请参阅 Amazon S3 基于身份的策略。

为了协助您浏览这些访问管理选项，以下是常见的 Amazon S3 客户使用案例和对于每个 S3 访问管理工具的建议。

AWS 账户拥有者希望仅与同一账户内的用户共享存储桶

所有访问管理工具都可以满足这个基本使用案例的要求。对于此使用案例，我们建议使用以下访问管理工具：

- 存储桶策略 – 如果您要授予对一个存储桶或少量存储桶的访问权限，或者如果不同存储桶之间的存储桶访问权限相似，请使用存储桶策略。使用存储桶策略，您可以为每个存储桶管理一个策略。有关更多信息，请参阅 Amazon S3 的存储桶策略。
- 基于身份的策略 – 如果您有大量的存储桶，每个存储桶的访问权限不同，并且只有几个用户角色需要管理，则可以对用户、组或角色使用 IAM 策略。如果您要管理用户对其它 AWS 资源以及

Amazon S3 资源的访问权限，IAM 策略也是一个不错的选择。有关更多信息，请参阅 示例 1：存储桶拥有者向其用户授予存储桶权限。

- S3 Access Grants – 您可以使用 S3 Access Grants 来授予对 S3 存储桶、前缀或对象的访问权限。S3 Access Grants 允许您大规模指定不同的对象级别权限，而存储桶策略的大小限制为 20 KB。有关更多信息，请参阅 开始使用 S3 Access Grants。
- 接入点 – 您可以使用接入点，接入点是附加到存储桶的命名网络端点。一个存储桶最多可附加 10000 个接入点，对于每个接入点，您可以强制实施不同的权限和网络控制，从而让您可以详细控制对于 S3 对象的访问权限。有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。

AWS 账户拥有者想要与其它 AWS 账户（跨账户）的用户共享存储桶或对象

要向其它 AWS 账户授予权限，您必须使用存储桶策略或以下建议的访问管理工具之一。您不能对此使用案例使用基于身份的访问策略。有关授予跨账户存取权限的更多信息，请参阅如何提供对 Amazon S3 存储桶中对象的跨账户存取权限？

对于此使用案例，我们建议使用以下访问管理工具：

- 存储桶策略 - 使用存储桶策略，您可以为每个存储桶管理一个策略。有关更多信息，请参阅 Amazon S3 的存储桶策略。
- S3 Access Grants – 您可以使用 S3 Access Grants 来授予对 S3 存储桶、前缀或对象的跨账户权限。您可以使用 S3 Access Grants 大规模指定不同的对象级别权限，而存储桶策略的大小限制为 20 KB。有关更多信息，请参阅 开始使用 S3 Access Grants。
- 接入点 – 您可以使用接入点，接入点是附加到存储桶的命名网络端点。一个存储桶最多可附加 10000 个接入点，对于每个接入点，您可以强制实施不同的权限和网络控制，从而让您可以详细控制对于 S3 对象的访问权限。有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。

AWS 账户拥有者或存储桶拥有者必须授予对象级别或前缀级别的权限，这些权限因对象或前缀而异

例如，在存储桶策略中，您可以授予对存储桶中共享特定 key name prefix 或具有特定标签的对象的访问权限。您可以授予对以键名称前缀 `logs/` 开头的对象的读取权限。但是，如果您的访问权限因对象而异，那么使用存储桶策略授予对各个对象的权限可能不太实际，特别是由于存储桶策略的大小限制为 20 KB。

对于此使用案例，我们建议使用以下访问管理工具：

- S3 Access Grants – 您可以使用 S3 Access Grants 来管理对象级别或前缀级别的权限。与存储桶策略不同，您可以使用 S3 Access Grants 大规模指定不同的对象级别权限。存储桶策略的大小限制为 20 KB。有关更多信息，请参阅 开始使用 S3 Access Grants。

- 接入点 - 您可以使用接入点来管理对象级别或前缀级别的权限。接入点是附加到存储桶的命名网络端点。一个存储桶最多可附加 10000 个接入点，对于每个接入点，您可以强制实施不同的权限和网络控制，从而让您可以详细控制对于 S3 对象的访问权限。有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。
- ACL – 我们建议不要使用访问控制列表（ACL），特别是因为 ACL 限制为每个对象 100 个授权。但是，如果您选择开启 ACL，请在存储桶设置中，将对象所有权 设置为存储桶拥有者优先 且 ACL 已启用。有了这个设置，用 `bucket-owner-full-control` 标准的 ACL 编写的新对象将自动归存储桶拥有者而不是对象编写者所有。然后，您可以使用对象 ACL（一个采用 XML 格式的访问策略）来向其他用户授予对于对象的访问权限。有关更多信息，请参阅 访问控制列表 (ACL) 概述。

AWS 账户拥有者或存储桶拥有者希望将存储桶访问权限限制为仅限特定的账户 ID

对于此使用案例，我们建议使用以下访问管理工具：

- 存储桶策略 - 使用存储桶策略，您可以为每个存储桶管理一个策略。有关更多信息，请参阅 Amazon S3 的存储桶策略。
- 接入点 – 接入点是附加到存储桶的命名网络端点。一个存储桶最多可附加 10000 个接入点，对于每个接入点，您可以强制实施不同的权限和网络控制，从而让您可以详细控制对于 S3 对象的访问权限。有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。

AWS 账户拥有者或存储桶拥有者希望每个访问其数据的用户或应用程序都使用不同的端点

对于此使用案例，我们建议使用以下访问管理工具：

- 接入点 – 接入点是附加到存储桶的命名网络端点。一个存储桶最多可附加 10000 个接入点，对于每个接入点，您可以强制实施不同的权限和网络控制，从而让您可以详细控制对于 S3 对象的访问权限。每个接入点强制实施自定义接入点策略，该策略与附加到底层存储桶的存储桶策略结合使用。有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。

AWS 账户拥有者或存储桶拥有者必须管理从 S3 的虚拟私有云（VPC）端点进行的访问

Amazon S3 的虚拟私有云（VPC）端点是 VPC 内仅允许连接到 S3 的逻辑实体。对于此使用案例，我们建议使用以下访问管理工具：

- VPC 中的存储桶设置 – 您可以使用存储桶策略来控制允许谁访问您的存储桶，以及他们可以访问哪些 VPC 端点。有关更多信息，请参阅 使用存储桶策略控制从 VPC 端点的访问。

- 接入点 - 如果您选择设置接入点，则可以使用接入点策略。您可以将任何接入点配置为仅接受来自 Virtual Private Cloud（VPC）的请求，以限制专用网络的 Amazon S3 数据访问。您还可以为每个接入点配置自定义屏蔽公共访问权限设置。有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。

AWS 账户拥有者或存储桶拥有者必须公开静态网站

使用 S3，您可以托管静态网站，并允许任何人查看该网站的内容（托管在 S3 存储桶中）。

对于此使用案例，我们建议使用以下访问管理工具：

- Amazon CloudFront – 该解决方案允许您面向公众托管 Amazon S3 静态网站，同时还继续屏蔽对存储桶内容的所有公共访问权限。如果要将所有四个“S3 屏蔽公共访问权限”设置保持为启用状态并托管 S3 静态网站，则可以使用 Amazon CloudFront 来源访问控制（OAC）。Amazon CloudFront 提供了设置安全静态网站所需的功能。此外，不使用此解决方案的 Amazon S3 静态网站只能支持 HTTP 端点。CloudFront 使用 Amazon S3 的持久存储，同时提供额外的安全标头，如 HTTPS。HTTPS 通过加密正常 HTTP 请求并防范常见的网络攻击来增强安全性。

  有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的安全静态网站入门。
- 让 Amazon S3 存储桶可供公开访问 – 您可以配置存储桶来用作公开访问的静态网站。

> **Warning**
>
> 我们建议不要使用这种方法。相反，我们建议您将 Amazon S3 静态网站用作 Amazon CloudFront 的一部分。有关更多信息，请参阅前一个选项或参阅安全静态网站入门。

要在没有 Amazon CloudFront 的情况下创建 Amazon S3 静态网站，首先必须关闭所有屏蔽公共访问权限设置。在为静态网站编写存储桶策略时，请确保仅允许 `s3:GetObject` 操作，而非 `ListObject` 或 `PutObject` 权限。这有助于确保用户无法查看存储桶中的所有对象或添加其自己的内容。有关更多信息，请参阅 设置访问网站的权限。

AWS 账户拥有者或存储桶拥有者想要公开存储桶的内容

创建新的 Amazon S3 存储桶时，屏蔽公共访问权限 设置默认处于启用状态。有关屏蔽公共访问权限的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

我们建议不要对存储桶允许公共访问权限。但是，如果您必须针对特定使用案例允许公共访问权限，我们建议您为此使用案例使用以下访问管理工具：

- 禁用屏蔽公共访问权限设置 - 存储桶拥有者可以允许向存储桶发送未经身份验证的请求。例如，当存储桶具有公有存储桶策略或存储桶 ACL 授予公共访问权限时，允许未经身份验证的 PUT Object 请求。所有未经身份验证的请求均由其他任意 AWS 用户发出，甚至是由未经身份验证的匿名用户发出。此用户在 ACL 中由特定的规范用户 ID `65a011a29cdf8ec533ec3d1ccaae921c` 表示。如果将对象上传到 `WRITE` 或 `FULL_CONTROL`，这会专门向“所有用户”组或匿名用户授予访问权限。有关公有存储桶策略和公有访问控制列表（ACL）的更多信息，请参阅 “公有”的含义。

AWS 账户拥有者或存储桶拥有者已超过访问策略大小限制

存储桶策略和基于身份的策略都有 20 KB 的大小限制。如果您的访问权限要求很复杂，则可能会超过此大小限制。

对于此使用案例，我们建议使用以下访问管理工具：

- 接入点 – 如果接入点适用于您的使用案例，请使用接入点。对于接入点，每个存储桶都有多个命名网络端点，每个端点都有自己的接入点策略，该策略与底层存储桶策略结合使用。但是，接入点只能对于对象执行操作，而不能对存储桶执行操作，并且不支持跨区域复制。有关更多信息，请参阅 使用 Amazon S3 接入点管理数据访问。
- S3 Access Grants – 使用 S3 Access Grants，这支持大量授权来授予对存储桶、前缀或对象的访问权限。有关更多信息，请参阅 开始使用 S3 Access Grants。

AWS 账户拥有者或管理员角色想要直接向公司目录中的用户或组授予存储桶、前缀或对象访问权限

您可以将公司目录添加到 AWS IAM Identity Center，而不必通过 AWS Identity and Access Management（IAM）管理用户、组和角色。有关更多信息，请参阅 What is IAM Identity Center?

将公司目录添加到 AWS IAM Identity Center 后，我们建议您使用以下访问管理工具，向公司目录身份授予对 S3 资源的访问权限：

- S3 Access Grants – 使用 S3 Access Grants，这支持向公司目录中的用户或角色授予访问权限。有关更多信息，请参阅 开始使用 S3 Access Grants。

AWS 账户拥有者或存储桶拥有者想要向 AWS CloudFront 服务授予访问权限，以便将 CloudFront 日志写入 S3 存储桶

对于此使用案例，我们建议使用以下访问管理工具：

- 存储桶 ACL – 存储桶 ACL 的唯一建议使用案例是向某些 AWS 服务（如 Amazon CloudFront `awslogsdelivery` 账户）授予权限。当您创建或更新分配并开启 CloudFront 日志记录时，CloudFront 会更新存储桶（ACL）来向 `awslogsdelivery` 账户授予 `FULL_CONTROL` 权限，从而将日志写入您的存储桶。有关更多信息，请参阅 Amazon CloudFront Developer Guide（Amazon CloudFront 开发人员指南）中的 Permissions required to configure standard logging and to access your log files（配置标准日志和访问日志文件所需的权限）。如果存储日志的存储桶对于 S3 对象所有权使用强制存储桶拥有者设置来关闭 ACL，则 CloudFront 无法将日志写入存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

作为存储桶拥有者，您希望对其他用户添加到存储桶中的对象保持完全控制权

您可以使用存储桶策略、接入点或 S3 Access Grants，来向其它账户授予将对象上传到您的存储桶的访问权限。如果您已授予对存储桶的跨账户存取权限，则可以确保上传到存储桶的任何对象均保持在您的完全控制之下。

对于此使用案例，我们建议使用以下访问管理工具：

- 对象所有权 - 将存储桶级别的设置对象所有权 保留为默认的强制存储桶拥有者 设置。

访问管理故障排除

以下资源有助于您排查 S3 访问管理方面的任何问题：

排查拒绝访问（403 禁止）错误

如果您遇到拒绝访问问题，请检查账户级别和存储桶级别的设置。此外，请检查您用于授予访问权限的访问管理功能，来确保策略、设置或配置正确无误。有关 Amazon S3 中拒绝访问（403 禁止）错误的常见原因的更多信息，请参阅排查 Amazon S3 中的拒绝访问（403 Forbidden）错误。

适用于 S3 的 IAM Access Analyzer

如果您不想公开您的任何资源，或者您想限制对资源的公共访问权限，则可以使用适用于 S3 的 IAM Access Analyzer。在 Amazon S3 控制台上，使用适用于 S3 的 IAM Access Analyzer 来查看具有存储桶访问控制列表（ACL）、存储桶策略或授予公共或共享访问权限的接入点策略的所有存储桶。如果存在已配置为允许互联网上的任何人或其他 AWS 账户（包括组织外部的 AWS 账户）访问的存储桶，适用于 S3 的 IAM Access Analyzer 会向您发出提醒。您会收到每个公共存储桶或共享存储桶的结果，其中报告了公共或共享访问的来源和级别。

在适用于 S3 的 IAM Access Analyzer 中，只需一个操作，即可屏蔽对存储桶的所有公共访问权限。我们建议您屏蔽所有对存储桶的公共访问权限，除非您需要公共访问权限才能支持特定使用案例。在屏蔽

所有公共访问权限之前，确保您的应用程序在没有公共访问权限的情况下可以继续正常工作。有关更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

您还可以查看存储桶级别的权限设置，来配置详细的访问级别。对于需要公共或共享访问的特定和经验证的使用案例，您可以通过对存储桶的结果进行归档来确认和记录存储桶保持公开或共享的意图。您可以随时重新访问和修改这些存储桶配置。您还可以将结果下载为 CSV 格式的报告以供审计使用。

Amazon S3 控制台上提供适用于 S3 的 IAM Access Analyzer，无需额外费用。适用于 S3 的 IAM Access Analyzer 由 AWS Identity and Access Management（IAM）IAM Access Analyzer 提供支持。要在 Amazon S3 控制台上使用适用于 S3 的 IAM Access Analyzer，您必须访问 IAM 控制台，然后在 IAM Access Analyzer 中为每个单独区域创建账户级别分析器。

有关适用于 S3 的 IAM Access Analyzer 的更多信息，请参阅使用适用于 S3 的 IAM Access Analyzer 查看存储桶访问权限。

日记账记录和监控

监控是维护 Amazon S3 解决方案的可靠性、可用性和性能的重要组成部分，以便您可以更轻松地调试访问故障。日志记录有助于您深入了解用户收到的任何错误，以及何时发出了什么样的请求。AWS 提供了几种用于监控 Amazon S3 资源的工具，如下所示：

- AWS CloudTrail
- Amazon S3 访问日志
- AWS Trusted Advisor
- Amazon CloudWatch

有关更多信息，请参阅 Amazon S3 中的日志记录和监控。

主题

- Amazon S3 的身份和访问管理
- 使用 S3 Access Grants 管理访问权限
- 使用 ACL 管理访问
- 阻止对您的 Amazon S3 存储的公有访问
- 使用适用于 S3 的 IAM Access Analyzer 查看存储桶访问权限
- 使用存储桶拥有者条件验证存储桶所有权
- 为您的存储桶控制对象所有权和禁用 ACL。

Amazon S3 的身份和访问管理

AWS Identity and Access Management（IAM）是一项 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）使用 Amazon S3 资源。IAM 是一项无需额外费用即可使用的 AWS 服务。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

主题

受众

如何使用 AWS Identity and Access Management（IAM）因您在 Amazon S3 中执行的操作而异。

服务用户：如果您使用 Amazon S3 服务来完成任务，管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon S3 功能来完成任务，您可能需要额外权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amazon S3 中的功能，请参阅Amazon S3 身份和访问问题排查。

服务管理员：如果您在公司负责管理 Amazon S3 资源，您可能对 Amazon S3 具有完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon S3 功能和资源。然后，您必须向 IAM 管理员提交请求

以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon S3 搭配使用的更多信息，请参阅Amazon S3 如何与 IAM 配合使用。

IAM 管理员：如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amazon S3 的访问的详细信息。要查看您可在 IAM 中使用的 Amazon S3 基于身份的策略示例，请参阅Amazon S3 基于身份的策略。

使用身份进行身份验证

身份验证是您使用身份凭证登录 AWS 的方法。您必须作为 AWS 账户根用户、IAM 用户或通过代入 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center（IAM Identity Center）用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合身份验证访问 AWS 时，您就是在间接代入角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录到 AWS 的更多信息，请参阅《AWS 登录 用户指南》中的如何登录到您的 AWS 账户。

如果您以编程方式访问 AWS，则 AWS 将提供软件开发工具包 (SDK) 和命令行界面 (CLI)，以便使用您的凭证以加密方式签署您的请求。如果您不使用 AWS 工具，则必须自行对请求签名。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，AWS 建议您使用多重身份验证（MFA）来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的在 AWS 中使用多重身份验证（MFA）。

AWS 账户 根用户

当您创建 AWS 账户 时，最初使用的是一个对账户中所有 AWS 服务 和资源拥有完全访问权限的登录身份。此身份称为 AWS 账户 根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）结合使用联合身份验证和身份提供程序，以使用临时凭证来访问 AWS 服务。

联合身份是来自企业用户目录、Web 身份提供程序、AWS Directory Service、Identity Center 目录的用户，或任何使用通过身份源提供的凭证来访问 AWS 服务 的用户。当联合身份访问 AWS 账户 时，他们代入角色，而角色提供临时凭证。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和组，也可以连接并同步到您自己的身份源中的一组用户和组以跨所有 AWS 账户 和应用程序使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？

IAM 用户和群组

IAM 用户是 AWS 账户 内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是 AWS 账户 中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS Management Console 中暂时代入 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义网址以担任角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。

- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。
- 跨服务访问：某些 AWS 服务使用其它 AWS 服务中的特征。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service (Amazon S3) 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话：当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务 委派权限的角色。
  - 服务相关角色 – 服务相关角色是与 AWS 服务 关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 AWS 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 AWS 身份或资源，以控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 AWS Management Console、AWS CLI 或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到 AWS 账户 中的多个用户、组和角色的独立策略。托管式策略包括 AWS 托管式策略和客户管理型策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 AWS 托管策略。

访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3、AWS WAF 和 VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的访问控制列表（ACL）概览。

其他策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单元 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）的权限。有关 Organizations 和 SCP 的更多信息，请参阅 AWS Organizations 用户指南中的服务控制策略。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 AWS 如何确定在涉及多种策略类型时是否允许请求，请参阅《IAM 用户指南》中的策略评估逻辑。

Amazon S3 如何与 IAM 配合使用

在使用 IAM 管理对 Amazon S3 的访问权限之前，您应该了解哪些 IAM 功能可用于 Amazon S3。

可以与 Amazon S3 配合使用的 IAM 功能

IAM 功能	Amazon S3 支持
基于身份的策略	是
基于资源的策略	是
策略操作	是
策略资源	是
策略条件键（特定于服务）	是

IAM 功能	Amazon S3 支持
ACL	是
ABAC（策略中的标签）	部分
临时凭证	是
转发访问会话（FAS）	是
服务角色	是
服务相关角色	部分

要深入了解 Amazon S3 和其它 AWS 服务如何与大多数 IAM 功能配合使用，请参阅《IAM 用户指南》中的使用 IAM 的 AWS 服务。

Amazon S3 基于身份的策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

Amazon S3 基于身份的策略示例

要查看 Amazon S3 基于身份的策略的示例，请参阅 Amazon S3 基于身份的策略。

Amazon S3 内基于资源的策略

支持基于资源的策略：是

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件

下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户存取，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当主体和资源处于不同的 AWS 账户 中时，则信任账户中的 IAM 管理员还必须授予主体实体（用户或角色）对资源的访问权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

Amazon S3 服务支持存储桶策略、接入点策略和访问授权：

- 存储桶策略是附加到 Amazon S3 存储桶的基于资源的策略。存储桶策略定义了哪些主体可以对该存储桶执行操作。
- 接入点策略是与底层存储桶策略一起进行评估的基于资源的策略。
- Access Grants 提供了一个简化模型，用于按前缀、存储桶或对象定义对 Amazon S3 中数据的访问权限。有关 S3 Access Grants 的信息，请参阅使用 S3 Access Grants 管理访问权限。

存储桶策略的主体

`Principal` 元素用于指定被允许或拒绝访问资源的用户、账户、服务或其它实体。下面是指定 `Principal` 的示例。有关更多信息，请参阅 IAM 用户指南中的主体。

向 AWS 账户授予权限

要向 AWS 账户授予权限，则使用以下格式标识此账户。

```
"AWS":"account-ARN"
```

示例如下。

```
"Principal":{"AWS":"arn:aws:iam::AccountIDWithoutHyphens:root"}
```

```
"Principal":{"AWS":
["arn:aws:iam::AccountID1WithoutHyphens:root","arn:aws:iam::AccountID2WithoutHyphens:root"]}
```

向 IAM 用户授予权限

要向您的账户中的 IAM 用户授予权限，您必须提供 "AWS":"*user-ARN*" 名称值对。

```
"Principal":{"AWS":"arn:aws:iam::account-number-without-hyphens:user/username"}
```

有关提供分步说明的详细示例，请参阅 示例 1：存储桶拥有者向其用户授予存储桶权限 和 示例 3：存储桶拥有者授予不属于自己的对象的权限。

> **Note**
>
> 如果您在更新存储桶策略后删除了 IAM 身份，则存储桶策略将在主体元素中显示唯一标识符，而不是 ARN。这些唯一 ID 从不会重复使用，因此，您可以安全地从所有策略语句中删除具有唯一标识符的主体。有关唯一标识符的更多信息，请参阅《IAM 用户指南》中的 IAM 标识符。

授予匿名权限

> **Warning**
>
> 在授予对您的 Amazon S3 存储桶的匿名访问权限时应谨慎。如果您授予匿名访问权限，那么世界上的任何人都可以访问您的存储桶。我们强烈建议您绝对不要授予对 S3 存储桶的任何类型的匿名写入权限。

要授予每个人权限，也称为匿名访问，请将 `Principal` 值设置为通配符 (`"*"`)。例如，如果您将存储桶配置为网站，您将需要使该存储桶中的所有对象都可公开访问。

```
"Principal":"*"
```

```
"Principal":{"AWS":"*"}
```

在基于资源的策略中使用具有 `Allow` 效果的 `"Principal": "*"` 允许任何人访问您的资源，即便未登录 AWS。

在基于资源的策略中使用具有 `Allow` 效果的 `"Principal" : { "AWS" : "*" }` 允许任何根用户、IAM 用户、代入角色会话或联合身份用户在同一分区的任何账户中访问您的资源。

对于匿名用户，这两种方法是等效的。有关更多信息，请参阅《IAM 用户指南》中的所有主体。

您不能使用通配符匹配一部分主体名称或 ARN。

> **Important**
>
> 因为任何人都可以创建 AWS 账户，安全级别在这两种方法中是等效的，尽管它们的功能不同。

限制资源权限

您还可以使用资源策略来限制对 IAM 主体原本可以使用的资源的访问权限。使用 Deny 语句来阻止访问。

如果未使用安全传输协议，则以下示例会阻止访问：

```
{"Effect": "Deny",
 "Principal": "*",
 "Action": "s3:*",
 "Resource": <bucket ARN>,
 "Condition": {
   "Boolean": { "aws:SecureTransport" : "false"}
 }
}
```

使用 "Principal": "*" 以便此限制适用于每个用户，这是本策略的最佳实践，而不是尝试使用此方法仅拒绝特定账户或主体的访问权限。

要求通过 CloudFront URL 进行访问

您可以要求用户仅使用 CloudFront URL 而不是 Amazon S3 URL 访问您的 Amazon S3 内容。为此，请创建 CloudFront 来源访问控制（OAC）。然后，更改 S3 数据的权限。在存储桶策略中，您可以将 CloudFront 设置为主体，如下所示：

```
"Principal":{"Service":"cloudfront.amazonaws.com"}
```

使用策略中的 Condition 元素，仅在请求代表包含 S3 源的 CloudFront 分配时，才允许 CloudFront 访问存储桶。

```
        "Condition": {
           "StringEquals": {
              "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/CloudFront-distribution-ID"
```

```
        }
    }
```

有关需要通过 CloudFront URL 访问 S3 的更多信息，请参阅《Amazon CloudFront 开发人员指南》中的限制对 Amazon Simple Storage Service 源的访问。有关使用 Amazon CloudFront 的安全和隐私优势的更多信息，请参阅配置安全访问和限制对内容的访问。

适用于 Amazon S3 的基于资源的策略示例

- 要查看 Amazon S3 存储桶的策略示例，请参阅Amazon S3 的存储桶策略。
- 要查看接入点的策略示例，请参阅配置使用接入点的 IAM 策略。

Amazon S3 的策略操作

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

下图显示了 S3 API 操作和所需策略操作之间不同类型的映射关系。

- 同名一对一映射。例如，要使用 `PutBucketPolicy` API 操作，需要执行 `s3:PutBucketPolicy` 策略操作。
- 具有不同名称的一对一映射。例如，要使用 `ListObjectsV2` API 操作，需要执行 `s3:ListBucket` 策略操作。
- 一对多映射。例如，要使用 `HeadObject` API 操作，则需要执行 `s3:GetObject`。此外，当您使用 S3 对象锁定并想要获取对象的法定保留状态或保留设置时，还需要执行相应的 `s3:GetObjectLegalHold` 或 `s3:GetObjectRetention` 策略操作，然后才能使用 `HeadObject` API 操作。
- 多对一映射。例如，要使用 `ListObjectsV2` 或 `HeadBucket` API 操作，需要执行 `s3:ListBucket` 策略操作。

要查看在策略中使用的 Amazon S3 操作的列表，请参阅《Service Authorization Reference》中的 Actions defined by Amazon S3。有关 Amazon S3 API 操作的完整列表，请参阅《Amazon Simple Storage Service API Reference》中的 Amazon S3 API Actions。

Amazon S3 中的策略操作在操作前面使用以下前缀：

```
s3
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "s3:action1",
      "s3:action2"
         ]
```

存储桶操作

存储桶操作是在存储桶资源类型上执行的 S3 API 操作。例如：`CreateBucket`、`ListObjectsV2` 和 `PutBucketPolicy`。针对存储桶操作的 S3 策略操作要求存储桶策略或基于 IAM 身份的策略中的 `Resource` 元素必须是采用以下示例格式的 S3 存储桶类型 Amazon 资源名称 (ARN) 标识符。

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
```

以下存储桶策略向账户 *12345678901* 的用户 *Akua* 授予执行 ListObjectsV2 API 操作和列出 S3 存储桶中对象的 `s3:ListBucket` 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
```

```
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
        }
    ]
}
```

接入点策略中的存储桶操作

在接入点策略中授予的权限仅在底层存储桶支持相同的权限时才有效。使用 S3 接入点时，必须将访问控制权从存储桶委托给接入点，或者将接入点策略中的相同权限添加到底层存储桶策略中。有关更多信息，请参阅 配置使用接入点的 IAM 策略。在接入点策略中，针对存储桶操作的 S3 策略操作要求您对 Resource 元素使用以下格式的 accesspoint ARN。

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT"
```

以下接入点策略向账户 *12345678901* 的用户 *Akua* 授予通过 S3 接入点 *DOC-EXAMPLE-ACCESS-POINT* 执行 ListObjectsV2 API 操作，以列出接入点关联存储桶中对象的 s3:ListBucket 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-
ACCESS-POINT"
        }
    ]
}
```

> **Note**
>
> 并非所有存储桶操作都受 S3 接入点支持。有关更多信息，请参阅 接入点与 S3 操作的兼容性。

对象操作

对象操作是按对象资源类型执行的 S3 API 操作。例如：GetObject、PutObject 和 DeleteObject。用于对象操作的 S3 策略操作要求策略中的 Resource 元素为采用以下示例格式的 S3 对象 ARN。

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/*"
```

> **Note**
>
> 对象 ARN 必须在存储桶名称后面包含正斜杠，如前面的示例所示。

以下存储桶策略向账户 *12345678901* 的用户 *Akua* 授予执行 PutObject API 操作以将对象上传到 S3 存储桶的 s3:PutObject 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to upload objects",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

接入点策略中的对象操作

使用 S3 接入点控制对对象操作的访问时，可以使用接入点策略。当您使用接入点策略时，用于对象操作的 S3 策略操作要求您对 Resource 元素使用以下格式的 accesspoint ARN：arn:aws:s3:region:account-id:accesspoint/access-point-name/object/

resource。对于使用接入点的对象操作，必须在 Resource 元素中整个接入点 ARN 的后面包含 /object/ 值。下面是一些示例。

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/object/*"
```

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/object/prefix/*"
```

以下接入点策略向账户 *12345678901* 的用户 *Akua* 授予通过接入点 *DOC-EXAMPLE-ACCESS-POINT* 对接入点关联存储桶中的所有对象执行 GetObject API 操作的 s3:GetObject 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to get objects through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": [
            "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT/object/*"
        }
    ]
}
```

> **Note**
>
> 并非所有对象操作都受 S3 接入点的支持。有关更多信息，请参阅 接入点与 S3 操作的兼容性。

接入点操作

接入点操作是在 accesspoint 资源类型上执行的 S3 API 操作。例如：CreateAccessPoint、DeleteAccessPoint 和 GetAccessPointPolicy。用于接入点操作

的 S3 策略操作只能在基于 IAM 身份的策略中使用，不能用于存储桶策略或接入点策略。接入点操作要求 Resource 元素必须是采用以下示例格式的 accesspoint ARN。

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/DOC-EXAMPLE-ACCESS-POINT"
```

以下基于 IAM 身份的策略授予在 S3 接入点 *DOC-EXAMPLE-ACCESS-POINT* 上执行 GetAccessPointPolicy API 操作的 s3:GetAccessPointPolicy 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Grant permission to retrieve the access point policy of access
 point DOC-EXAMPLE-ACCESS-POINT",
            "Effect": "Allow",
            "Action": [
            "s3:GetAccessPointPolicy"
            ],
            "Resource": "arn:aws:s3:*:123456789012:access point/DOC-EXAMPLE-ACCESS-
POINT"
        }
    ]
}
```

使用接入点时，要控制对存储桶操作的访问权限，请参阅接入点策略中的存储桶操作；要控制对对象操作的访问权限，请参阅接入点策略中的对象操作。有关如何配置接入点策略的更多信息，请参阅配置使用接入点的 IAM 策略。

对象 Lambda 接入点操作

借助 Amazon S3 对象 Lambda，您可以将自己的代码添加到 Amazon S3 GET、LIST 和 HEAD 请求中，以便在数据返回到应用程序时修改和处理数据。您可以通过 Object Lambda 接入点发出请求，与通过其它接入点发出请求的工作方式相同。有关更多信息，请参阅 使用 S3 对象 Lambda 转换对象。

有关如何为对象 Lambda 接入点操作配置策略的更多信息，请参阅为对象 Lambda 接入点配置 IAM 策略。

多区域接入点操作

多区域接入点提供了一个全局端点，应用程序可以使用该端点来满足来自位于多个 AWS 区域中的 S3 存储桶的请求。您可以使用多区域接入点通过单个区域中使用的相同架构来构建多区域应用程序，然后在世界任何地方运行这些应用程序。有关更多信息，请参阅 Amazon S3 中的多区域接入点。

有关如何为多区域接入点操作配置策略的更多信息，请参阅多区域接入点策略示例。

批处理作业操作

（批量操作）作业操作是在作业资源类型上执行的 S3 API 操作。例如，DescribeJob 和 CreateJob。用于作业操作的 S3 策略操作只能在基于 IAM 身份的策略中使用，不能用于存储桶策略。此外，作业操作要求基于 IAM 身份的策略中的 Resource 元素必须是采用以下示例格式的 job ARN。

```
"Resource": "arn:aws:s3:*:123456789012:job/*"
```

以下基于 IAM 身份的策略授予对 S3 Batch Operations 作业 *DOC-EXAMPLE-JOB* 执行 DescribeJob API 操作的 s3:DescribeJob 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow describing the Batch operation job DOC-EXAMPLE-JOB",
            "Effect": "Allow",
            "Action": [
            "s3:DescribeJob"
            ],
            "Resource": "arn:aws:s3:*:123456789012:job/DOC-EXAMPLE-JOB"
        }
    ]
}
```

S3 Storage Lens 存储分析功能配置操作

有关如何配置 S3 Storage Lens 存储分析功能配置操作的更多信息，请参阅Amazon S3 Storage Lens 存储统计管理工具权限。

账户操作

账户操作是在账户级别执行的 S3 API 操作。例如，GetPublicAccessBlock（用于账户）。账户不是 Amazon S3 定义的资源类型。用于账户操作的 S3 策略操作只能在基于 IAM 身份的策略中使用，不能用于存储桶策略。此外，账户操作要求基于 IAM 身份的策略中的 Resource 元素必须是 "*"。

以下基于 IAM 身份的策略授予执行账户级别 GetPublicAccessBlock API 操作和检索账户级公共访问权限屏蔽设置的 s3:GetAccountPublicAccessBlock 权限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"Allow retrieving the account-level Public Access Block settings",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

Amazon S3 的策略示例

- 要查看 Amazon S3 基于身份的策略的示例，请参阅 Amazon S3 基于身份的策略。
- 要查看 Amazon S3 基于资源的策略示例，请参阅Amazon S3 的存储桶策略和配置使用接入点的 IAM 策略。

Amazon S3 的策略资源

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon 资源名称 (ARN) 指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

某些 Amazon S3 API 操作支持多个资源。例如，`s3:GetObject` 访问 EXAMPLE-RESOURCE-1 和 EXAMPLE-RESOURCE-2，因此主体必须具有访问这两个资源的权限。要在单个语句中指定多个资源，请使用逗号分隔 ARN。

```
"Resource": [
      "EXAMPLE-RESOURCE-1",
      "EXAMPLE-RESOURCE-2"
```

Amazon S3 中的资源包括存储桶、对象、接入点或任务。在策略中，使用存储桶、对象、接入点或任务的 Amazon 资源名称（ARN）来识别资源。

有关 Amazon S3 资源类型及其 ARN 的列表，请参阅《Service Authorization Reference》中的 Resources defined by Amazon S3。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Actions defined by Amazon S3。

资源 ARN 的通配符

您可将通配符作为资源 ARN 的一部分。您可在任何 ARN 分段 (用冒号分隔的部分) 中使用通配符 (* 和 ?)。星号 (*) 表示 0 个或多个字符的任意组合，问号 (?) 表示任何单个字符。您可在每个分段中使用多个 * 或 ? 字符，但通配符不能跨分段。

- 以下 ARN 在此 ARN 的相对 ID 部分使用通配符 * 来标识 examplebucket 存储桶中的所有对象。

  ```
  arn:aws:s3:::examplebucket/*
  ```

- 以下 ARN 使用 * 来表示所有 S3 存储桶和对象。

  ```
  arn:aws:s3:::*
  ```

- 以下 ARN 在 relative-ID 部分中同时使用通配符 * 和 ?。它标识存储桶（例如 example1bucket、example2bucket、example3bucket 等）中的所有对象。

  ```
  arn:aws:s3:::example?bucket/*
  ```

资源 ARN 的策略变量

您可以在 Amazon S3 ARN 中使用策略变量。在策略评估时，这些预定义变量被它们的相应值替换。假设您将存储桶组织为一个文件夹的集合，每个用户拥有一个文件夹。文件夹名称与用户名称相同。要为他们的文件夹授予用户权限，您可以在资源 ARN 中指定策略变量：

```
arn:aws:s3:::bucket_name/developers/${aws:username}/
```

在运行时，如果评估策略，资源 ARN 中的 `${aws:username}` 变量将替换为发出请求的人员的用户名。

Amazon S3 的策略示例

- 要查看 Amazon S3 基于身份的策略的示例，请参阅 Amazon S3 基于身份的策略。
- 要查看 Amazon S3 基于资源的策略示例，请参阅Amazon S3 的存储桶策略和配置使用接入点的 IAM 策略。

Amazon S3 的策略条件键

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则 AWS 使用逻辑 OR 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

AWS 支持全局条件键和特定于服务的条件键。要查看所有 AWS 全局条件键，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

每个 Amazon S3 条件键均映射到可设置条件的 API 所允许的相同名称请求标头。特定于 Amazon S3 的条件键指定同名请求标头的行为。例如，条件键 `s3:VersionId` 用于对 `s3:GetObjectVersion` 权限授予条件权限，它定义您在 GET Object 请求中设置的查询参数 `versionId` 的行为。

要查看 Amazon S3 条件键的列表，请参阅《Service Authorization Reference》中的 Condition keys for Amazon S3。要了解您可以对哪些操作和资源使用条件键，请参阅 Actions defined by Amazon S3。

示例：将对象上传限制为具有特定存储类的对象

假设账户 A（由账户 ID 123456789012 表示）拥有一个存储桶。账户 A 管理员想要限制 Dave（账户 A 中的一名用户）只能将使用 STANDARD_IA 存储类存储的对象上传到存储桶。要将对象上传限制到特定的存储类，账户 A 管理员可以使用 s3:x-amz-storage-class 条件键，如以下示例存储桶策略所示。

```
{
                "Version": "2012-10-17",
                "Statement": [
                  {
                    "Sid": "statement1",
                    "Effect": "Allow",
                    "Principal": {
                      "AWS": "arn:aws:iam::123456789012:user/Dave"
                    },
                    "Action": "s3:PutObject",
                    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                    "Condition": {
                      "StringEquals": {
                        "s3:x-amz-storage-class": [
                          "STANDARD_IA"
                        ]
                      }
                    }
                  }
                ]
            }
```

在本例中，Condition 块指定应用于指定的键值对 "s3:x-amz-acl":["public-read"] 的 StringEquals 条件。有一组预定义键可用于表达条件。此示例使用 s3:x-amz-acl 条件键。此条件要求用户包含 x-amz-acl 标头，并且值 public-read 位于每个 PUT 对象请求中。

Amazon S3 的策略示例

- 要查看 Amazon S3 基于身份的策略的示例，请参阅 Amazon S3 基于身份的策略。
- 要查看 Amazon S3 基于资源的策略示例，请参阅Amazon S3 的存储桶策略和配置使用接入点的 IAM 策略。

Amazon S3 中的 ACL

支持 ACL：是

在 Amazon S3 中，访问控制列表（ACL）控制哪些 AWS 账户有权限访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

> **Important**
>
> Amazon S3 中的大多数现代使用案例不再需要使用 ACL。

有关如何使用 ACL 在 Amazon S3 中控制访问的信息，请参阅使用 ACL 管理访问。

ABAC 与 Amazon S3

支持 ABAC（策略中的标签）：部分支持

基于属性的访问控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在 AWS 中，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）以及 AWS 资源。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`* 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

要查看基于身份的策略（用于根据标签来限制对 S3 分批操作的访问）的示例，请参阅使用任务标签控制 S3 分批操作的权限。

ABAC 和对象标签

在 ABAC 策略中，对象使用 `s3:` 标签而不是 `aws:` 标签。要根据对象标签控制对对象的访问，您需要在策略的 condition 元素中使用以下标签提供标签信息。

- `s3:ExistingObjectTag/`*`tag-key`*

- `s3:s3:RequestObjectTagKeys`
- `s3:RequestObjectTag/tag-key`

有关使用对象标签控制访问的信息，包括权限策略示例，请参阅标签和访问控制策略。

将临时凭证用于 Amazon S3

支持临时凭证：是

某些 AWS 服务 在您使用临时凭证登录时无法正常工作。有关更多信息，包括 AWS 服务 与临时凭证配合使用，请参阅 IAM 用户指南中的使用 IAM 的 AWS 服务。

如果您不使用用户名和密码而用其它方法登录到 AWS Management Console，则使用临时凭证。例如，当您使用贵公司的单点登录（SSO）链接访问 AWS 时，该过程将自动创建临时凭证。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的切换到角色（控制台）。

您可以使用 AWS CLI 或者 AWS API 创建临时凭证。之后，您可以使用这些临时凭证访问 AWS。AWS 建议您动态生成临时凭证，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

Amazon S3 的转发访问会话

支持转发访问会话（FAS）：是

当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

- 当使用 SSE-KMS 加密对象时，Amazon S3 使用 FAS 调用 AWS KMS 以解密该对象。有关更多信息，请参阅 使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。
- S3 Access Grants 还使用 FAS。在您为特定身份创建针对 S3 数据的访问授权后，被授权者会向 S3 Access Grants 申请临时凭证。S3 Access Grants 会从 AWS STS 获取请求者的临时凭证，然后将该凭证发送给请求者。有关更多信息，请参阅 通过 S3 Access Grants 请求访问 Amazon S3 数据。

Amazon S3 的服务角色

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务 委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 Amazon S3 的功能。仅当 Amazon S3 提供相关指导时才编辑服务角色。

Amazon S3 的服务相关角色

支持服务相关角色：部分支持

服务相关角色是一种与 AWS 服务 相关的服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 AWS 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

Amazon S3 支持针对 Amazon S3 Storage Lens 存储统计管理工具的服务相关角色。有关创建或管理 Amazon S3 服务相关角色的详细信息，请参阅将服务相关角色用于 Amazon S3 Storage Lens 存储统计管理工具。

作为主体的 Amazon S3 服务

策略中的服务名称	S3 功能	更多信息
`s3.amazonaws.com`	S3 复制	设置实时复制
`s3.amazonaws.com`	S3 事件通知	Amazon S3 事件通知
`s3.amazonaws.com`	S3 清单	Amazon S3 清单
`access-grants.s3.amazonaws.com`	S3 访问授权	注册位置
`batchoperations.s3.amazonaws.com`	S3 批量操作	授予 Amazon S3 分批操作的权限
`logging.s3.amazonaws.com`	S3 服务器访问日志记录	启用 Amazon S3 服务器访问日志记录

策略中的服务名称	S3 功能	更多信息
`storage-lens.s3.amazonaws.com`	S3 Storage Lens 存储统计管理工具	使用数据导出查看 Amazon S3 Storage Lens 存储统计管理工具指标

Amazon S3 中的策略和权限

本页概述 Amazon S3 中的存储桶和用户策略，并介绍 AWS Identity and Access Management（IAM）策略的基本元素。每个列出的元素都链接到有关该元素以及如何使用它的示例的更多详细信息。

有关 Amazon S3 操作、资源和条件的完整列表，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

就其最基本的意义而言，策略包含以下元素：

- Resource – 策略适用的 Amazon S3 存储桶、对象、接入点或任务。使用存储桶、对象、接入点或任务的 Amazon 资源名称（ARN）来识别资源。

  存储桶级操作示例：

  `"Resource": "arn:aws:s3:::bucket_name"`

  对象级操作示例：

  - `"Resource": "arn:aws:s3:::bucket_name/*"` 对应于存储桶中的所有对象。
  - `"Resource": "arn:aws:s3:::bucket_name/prefix/*"` 对应于存储桶中特定前缀下的对象。

  有关更多信息，请参阅 Amazon S3 的策略资源。

- Actions – 对于每个资源，Amazon S3 支持一组操作。您可使用操作关键字标识允许（或拒绝）的资源操作。

  例如，`s3:ListBucket` 权限可让用户使用 Amazon S3 ListObjectsV2 操作。（`s3:ListBucket` 权限是指操作名称不直接映射到操作名称的情况。）有关使用 Simple Storage Service (Amazon S3) 操作的更多信息，请参阅 Amazon S3 的策略操作。有关Amazon S3 操作的完整列表，请参阅 Amazon Simple Storage Service API 参考中操作部分。

- Effect – 当用户请求特定操作（可以是 `Allow` 或 `Deny`）时的效果。

如果没有显式授予（允许）对资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问。您可以执行此操作以确保用户无法访问资源，即使其他策略授予访问权限也是如此。有关更多信息，请参阅《IAM 用户指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_effect.html中的 IAM JSON 策略元素：效果。

- Principal – 支持在语句中访问操作和资源的账户或用户。在存储桶策略中，主体是作为此权限的接收者的用户、账户、服务或其他实体。有关更多信息，请参阅 存储桶策略的主体。
- Condition – 策略生效时的条件。您可以使用 AWS 范围内的键和特定于 Amazon S3 的键来指定 Amazon S3 访问策略中的条件。有关更多信息，请参阅 使用条件键的存储桶策略示例。

下面的示例存储桶策略显示了 `Effect`、`Principal`、`Action` 和 `Resource` 元素。此策略可让账户 *123456789012* 中的用户 *Akua* 对 *amzn-s3-demo-bucket1* 存储桶具有 `s3:GetObject`、`s3:GetBucketLocation` 和 `s3:ListBucket` Amazon S3 权限。

```
{
    "Version": "2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Akua"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        }
    ]
}
```

有关完整的策略语言信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限和 IAM JSON 策略参考。

授予权限

如果某 AWS 账户拥有一项资源，它可以将这些权限授予其他 AWS 账户。然后这个账户就可以将这些权限或其子集委托给该账户中的用户。这称为 权限委托。但从另一账户接收权限的账户不能向其它 AWS 账户跨账户委托权限。

Amazon S3 存储桶和对象所有权

存储桶和对象是 Amazon S3 资源。默认情况下，只有资源所有者可以访问这些资源。资源拥有者是指创建资源的 AWS 账户。例如：

- 用于创建存储桶和上传对象的 AWS 账户拥有这些资源。
- 如果您使用 AWS Identity and Access Management (IAM) 用户或角色凭证上传对象，则用户或角色所属的 AWS 账户将拥有该对象。
- 存储桶拥有者可以向其他 AWS 账户（或其他账户中的用户）授予上传对象的跨账户权限。在这种情况下，上传对象的 AWS 账户拥有这些对象。存储桶拥有者对其它账户拥有的对象没有权限，以下情况除外：
  - 账单由存储桶拥有者支付。存储桶拥有者可以拒绝对任何对象的访问，或删除存储桶中的任何对象，而无论它们的拥有者是谁。
  - 存储桶拥有者可以归档任何对象或还原归档的对象，而无论它们的拥有者是谁。归档是指用于存储对象的存储类。有关更多信息，请参阅 管理存储生命周期。

所有权和请求身份验证

存储桶的所有请求已经过身份验证或未经过身份验证。经过身份验证的请求必须包含可验证请求发送者的签名值，而未经身份验证的请求则不然。有关请求身份验证的更多信息，请参阅 提出请求。

存储桶拥有者可以允许未经身份验证的请求。例如，当存储桶具有公有存储桶策略，或存储桶 ACL 特别向 `All Users` 组或匿名用户授予 `WRITE` 或 `FULL_CONTROL` 访问权限时，允许未经身份验证的 PutObject 请求。有关公有存储桶策略和公有访问控制列表（ACL）的更多信息，请参阅 “公有”的含义。

所有未经身份验证的请求均由匿名用户发起。此用户在 ACL 中由特定的规范用户 ID `65a011a29cdf8ec533ec3d1ccaae921c` 表示。如果对象是通过未经身份验证的请求上传到存储桶的，匿名用户拥有对象。默认对象 ACL 向作为对象拥有者的匿名用户授予 `FULL_CONTROL`。因此，Amazon S3 允许未经身份验证的请求检索该对象或修改其 ACL。

为防止匿名用户修改对象，建议您不要实施允许对存储桶进行匿名公有写入的存储桶策略，或不要使用允许存储桶的匿名用户写入权限的 ACL。您可以使用 Amazon S3 屏蔽公共访问权限强制实施这个建议行为。

有关屏蔽公共访问权限的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。有关 ACL 的更多信息，请参阅 访问控制列表 (ACL) 概述。

> **Important**
>
> 我们建议不要使用 AWS 账户根用户凭证发起经过身份验证的请求。而应创建一个 IAM 角色并授予该角色完全访问权限。我们将具有此角色的用户称为管理员用户。您可以使用分配给管理员角色的用户凭证而不是 AWS 账户 根用户凭证来与 AWS 交互和执行任务，例如创建存储桶、创建用户以及授予权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 安全凭证和 IAM 的安全防御最佳实操。

Amazon S3 的存储桶策略

存储桶策略是基于资源的策略，您可以使用该策略向 Amazon S3 存储桶及其中的对象授予访问权限。只有存储桶拥有者才能将策略与存储桶关联。附加到存储桶的权限适用于存储桶拥有者拥有的存储桶中所有对象。这些权限不适用于其它 AWS 账户所拥有的对象。

S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权并禁用或启用访问控制列表（ACL）。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。存储桶拥有者拥有存储桶中的所有对象，并使用策略专门管理对数据的访问权限。

存储桶策略使用基于 JSON 的 AWS Identity and Access Management（IAM）策略语言。您可以使用存储桶策略添加或拒绝存储桶中对象的权限。存储桶策略可以基于策略中的元素允许或拒绝请求。这些元素包括请求者、S3 操作、资源以及请求的特征或条件（例如，用于发出请求的 IP 地址）。

例如，您可以创建执行以下操作的存储桶策略：

- 授予其他账户将对象上传到您的 S3 存储桶的跨账户权限
- 确保您（存储桶拥有者）对于上传的对象具有完全控制权限

有关更多信息，请参阅 Amazon S3 存储桶策略的示例。

> **Important**
>
> 您不能使用存储桶策略来防止通过 S3 生命周期规则进行删除或转换。例如，即使您的存储桶策略拒绝所有主体的所有操作，您的 S3 生命周期配置也仍能正常发挥作用。

在本节中，我们通过主题提供了示例，并向您展示了如何在 S3 控制台中添加存储桶策略。有关基于身份的策略的信息，请参阅Amazon S3 基于身份的策略。有关存储桶策略语言的信息，请参阅Amazon S3 中的策略和权限。

主题

- 使用 Amazon S3 控制台添加存储桶策略
- 使用存储桶策略控制从 VPC 端点的访问
- Amazon S3 存储桶策略的示例
- 使用条件键的存储桶策略示例

使用 Amazon S3 控制台添加存储桶策略

您可以使用 AWS 策略生成器和 Amazon S3 控制台添加新的存储桶策略或编辑现有存储桶策略。存储桶策略是基于资源的 AWS Identity and Access Management（IAM）策略。将存储桶策略添加到一个存储桶可向其他 AWS 账户 或 IAM 用户授予对该存储桶以及其中对象的访问权限。对象权限仅应用于存储桶拥有者创建的对象。有关存储桶策略的更多信息，请参阅 Amazon S3 的身份和访问管理。

确保保存策略之前解决来自 AWS Identity and Access Management Access Analyzer 的安全警告、错误、一般警告和建议。IAM Access Analyzer 将根据 IAM 策略语法和最佳实践运行策略检查，以验证您的策略。这些检查项生成结果并提供可操作的建议，可帮助您编写可操作且符合安全最佳实践的策略。要了解有关使用 IAM Access Analyzer 验证策略的更多信息，请参阅《IAM 用户指南》中的 IAM Access Analyzer 策略验证。要查看 IAM Access Analyzer 返回的警告、错误和建议的列表，请参阅 IAM Access Analyzer 策略检查引用。

有关对策略错误进行故障排查的指南，请参阅排查 Amazon S3 中的拒绝访问（403 Forbidden）错误。

创建或编辑存储桶策略

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（存储桶）列表中，请选择要为其创建或编辑存储桶策略的存储桶的名称。
4. 选择 Permissions（权限）选项卡。
5. 在 Bucket policy（存储桶策略）下，请选择 Edit（编辑）。将出现 Edit bucket policy（编辑存储桶策略）页面。
6. 在 Edit bucket policy（编辑存储桶策略）页面上，执行以下操作之一：

   - 要查看存储桶策略的示例，请选择策略示例。或请参阅《Amazon S3 用户指南》中的 Amazon S3 存储桶策略的示例。
   - 要自动生成策略或编辑策略部分中的 JSON，请选择策略生成器。

   如果选择 Policy generator（策略生成器），AWS 策略生成器将在新窗口中打开。

   a. 在 AWS 策略生成器页面上，对于选择策略类型，选择 S3 存储桶策略。
   b. 通过在提供的字段中输入信息来添加语句，然后选择添加语句。对所有您想添加的语句重复执行此步骤。有关这些字段的更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

      > **Note**
      >
      > 为方便起见，编辑桶策略页面会在策略文本字段上方显示当前桶的桶 ARN（Amazon 资源名称）。您可以复制此 ARN，以便在 AWS 策略生成器页面上的语句中使用。

   c. 添加完语句后，请选择生成策略。
   d. 复制生成的策略文本，请选择 Close（关闭），然后返回到 Amazon S3 控制台中的 Edit bucket policy（编辑存储桶策略）页面。
7. 在 Policy（策略）框中，编辑现有策略或从 AWS 策略生成器粘贴存储桶策略。确保在保存策略之前解决安全警告、错误、一般警告和建议。

   > **Note**
   >
   > 存储桶策略的大小限制为 20 KB。

8. （可选）选择右下角的 Preview external access（预览外部访问），以预览新策略如何影响对资源的公有和跨账户访问。在保存策略之前，您可以检查策略是引入了新的 IAM Access Analyzer 发现结果还是解析了现有的发现结果。如果您没有看到活动的分析器，请在 IAM Access Analyzer 中

选择 Go to Access Analyzer（转至 Access Analyzer）以创建账户分析器。有关更多信息，请参阅 IAM 用户指南中的预览访问权限。

9. 请选择 Save changes（保存更改），此操作将让您返回到 Permissions（权限）选项卡。

使用存储桶策略控制从 VPC 端点的访问

您可以使用 Amazon S3 存储桶策略控制从特定虚拟私有云（VPC）端点或特定 VPC 对存储桶的访问。本部分包含可用于从 VPC 端点控制 Amazon S3 存储桶访问的示例存储桶策略。要了解如何设置 VPC 端点，请参阅 VPC 用户指南中的 VPC 端点。

您可以使用 VPC 在您定义的虚拟网络内启动 AWS 资源。使用 VPC 端点可以在您的 VPC 和其它 AWS 服务之间创建私有连接。此私有连接不需要通过互联网、虚拟专用网络（VPN）连接、NAT 实例或 AWS Direct Connect 进行访问。

Amazon S3 的 VPC 端点是 VPC 内的逻辑实体，仅允许连接到 Amazon S3。VPC 端点将请求路由到 Amazon S3 并将响应路由回 VPC。VPC 端点仅更改请求的路由方式。Amazon S3 公有端点和 DNS 名称将继续使用 VPC 端点。有关在 Amazon S3 中使用 VPC 端点的重要信息，请参阅《VPC 用户指南》中的 Gateway endpoints 和 Gateway endpoints for Amazon S3。

Amazon S3 的 VPC 端点提供两种方式来控制对 Amazon S3 数据的访问：

- 您可以控制允许通过特定 VPC 端点访问的请求、用户或组。有关此类型的访问控制的信息，请参阅《VPC 用户指南》中的 Controlling access to VPC endpoints using endpoint policies。
- 可以使用 Amazon S3 存储桶策略控制哪些 VPC 或 VPC 端点有权访问存储桶。有关此类型存储桶策略访问控制的示例，请参阅以下有关限制访问的主题。

主题

- 限制对特定 VPC 端点的访问
- 限制对特定 VPC 的访问

> **Important**
>
> 如本节中所述对 VPC 端点应用 Amazon S3 存储桶策略时，您可能会无意中阻止对存储桶的访问权限。存储桶权限旨在专门限制存储桶访问源自 VPC 端点的连接，而这可能会阻止到存储桶的所有连接。有关如何修复此问题的信息，请参阅《AWS Support 知识中心》中的当 VPC 或 VPC 端点 ID 错误时，如何修复存储桶策略？

限制对特定 VPC 端点的访问

下面是限制仅从 ID 为 awsexamplebucket1 的 VPC 端点访问特定存储桶 vpce-1a2b3c4d 的 Amazon S3 存储桶策略示例。如果未使用指定的端点，则该策略会拒绝对存储桶的所有访问。aws:SourceVpce 条件指定端点。aws:SourceVpce 条件不需要 VPC 端点资源的 Amazon 资源名称（ARN），而只需要 VPC 端点 ID。有关在策略中使用条件的更多信息，请参阅 使用条件键的存储桶策略示例。

> **Important**
>
> - 在使用以下示例策略之前，将 VPC 端点 ID 替换为适合您的使用案例的值。否则，您将无法访问您的存储桶。
> - 此策略禁用控制台访问指定的存储桶，因为控制台请求不是来自指定的 VPC 端点。

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909152",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPCE-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::awsexamplebucket1",
                    "arn:aws:s3:::awsexamplebucket1/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpce": "vpce-1a2b3c4d"
         }
       }
     }
   ]
}
```

限制对特定 VPC 的访问

可以使用 aws:SourceVpc 条件来创建用于限制对特定 VPC 的访问的存储桶策略。如果在同一 VPC 中配置了多个 VPC 端点，并且要管理对所有端点的 Amazon S3 存储桶的访问，这一点非常有用。以下是拒绝 VPC vpc-111bbb22 以外的任何人访问 awsexamplebucket1 及其对象的策略示例。如果

未使用指定的 VPC，则该策略拒绝对存储桶的所有访问。此语句不授予针对存储桶的访问权限。要授予访问权限，必须添加单独的 Allow 语句。vpc-111bbb22 条件键不需要 VPC 资源 的 ARN，仅需要 VPC ID。

> **Important**
>
> - 在使用以下示例策略之前，将 VPC ID 替换为适合您的使用案例的值。否则，您将无法访问您的存储桶。
> - 此策略禁用控制台访问指定的存储桶，因为控制台请求不是来自指定的 VPC。

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909153",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPC-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::awsexamplebucket1",
                    "arn:aws:s3:::awsexamplebucket1/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpc": "vpc-111bbb22"
         }
       }
     }
   ]
}
```

Amazon S3 存储桶策略的示例

使用 Amazon S3 存储桶策略，您可以保护对存储桶中对象的访问，这样，只有具有适当权限的用户才能访问它们。您甚至可以阻止没有适当权限的经过身份验证的用户访问您的 Amazon S3 资源。

本节介绍关于存储桶策略的典型使用案例的示例。这些示例策略将 *amzn-s3-demo-bucket* 用作资源值。要测试这些策略，您需要将 *user input placeholders* 替换为您自己的信息（例如存储桶名称）。

要授予或拒绝对一组对象的权限，可以在 Amazon资源名称（ARN）和其他值中使用通配符（*）。例如，您可以控制对以通用前缀或以特定扩展名结尾的对象组的访问，例如 .html。

有关 AWS Identity and Access Management（IAM）策略语言的更多信息，请参阅Amazon S3 中的策略和权限。

> **Note**
>
> 使用 Amazon S3 控制台测试权限时，您必须授予控制台所需的其他权限（s3:ListAllMyBuckets、s3:GetBucketLocation 和 s3:ListBucket）。有关向用户授予权限并使用控制台测试这些权限的示例演练，请参阅使用用户策略控制对存储桶的访问。

用于创建存储桶策略的其它资源包括以下内容：

- 有关在创建存储桶策略时可以使用的 IAM 策略操作、资源和条件键的列表，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。
- 有关创建 S3 策略的指南，请参阅使用 Amazon S3 控制台添加存储桶策略。
- 要对策略错误进行故障排查，请参阅排查 Amazon S3 中的拒绝访问（403 Forbidden）错误。

主题

- 向公共匿名用户授予只读权限
- 需要加密
- 使用标准 ACL 管理存储桶
- 使用对象标记管理对象访问权限
- 使用全局条件键管理对象访问权限
- 管理基于 HTTP 或 HTTPS 请求的访问权限
- 管理用户对特定文件夹的访问权限
- 管理访问日志的访问权限
- 管理对 Amazon CloudFront OAI 的访问
- 管理 Amazon S3 Storage Lens 存储统计管理工具的访问权限
- 管理 S3 清单、S3 分析和 S3 清单报告的权限

- 需要 MFA
- 防止用户删除对象

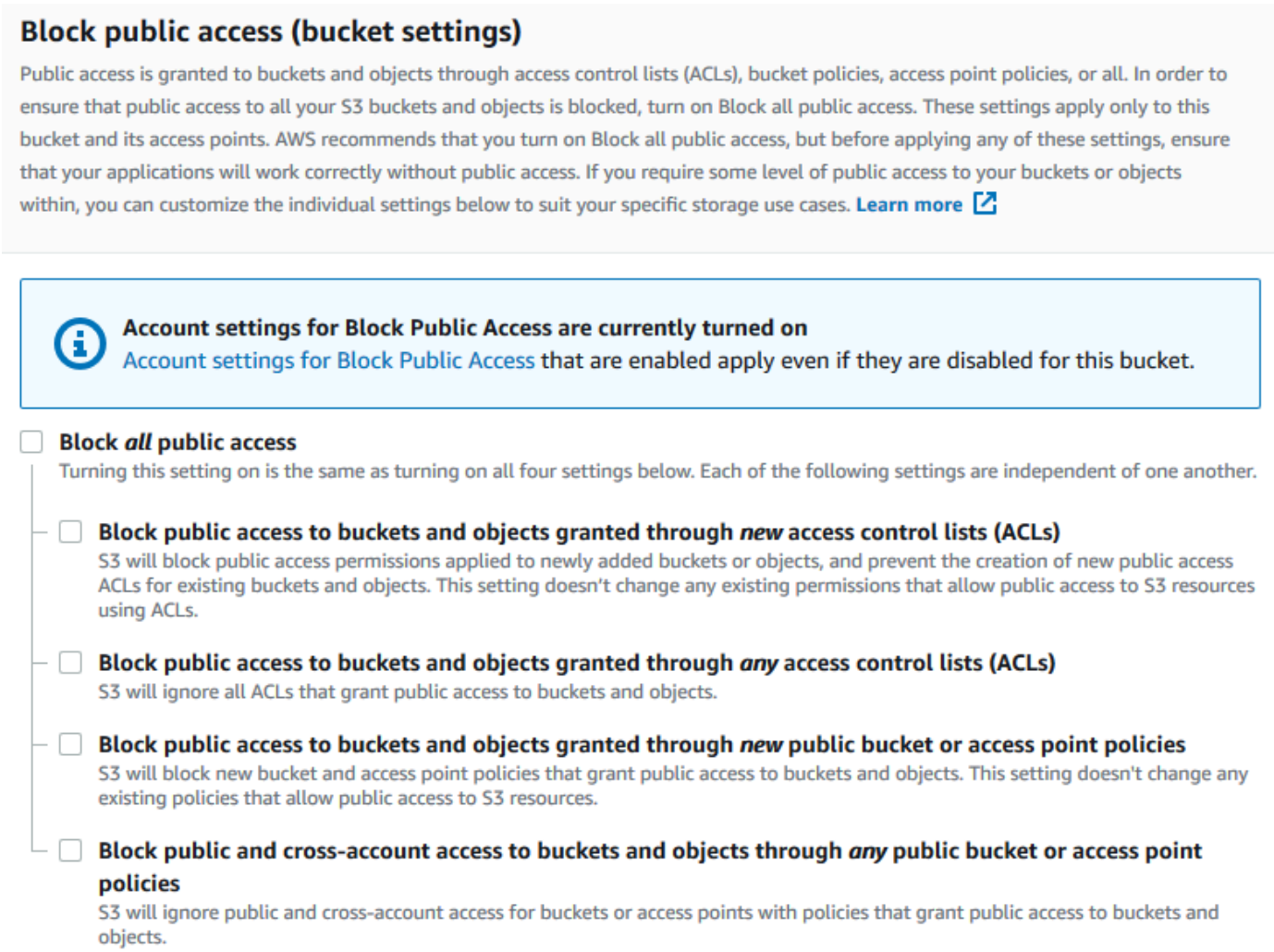
向公共匿名用户授予只读权限

可以使用策略设置向公共匿名用户授予访问权限，如果您要将存储桶配置为静态网站，这将非常有用。向公共匿名用户授予访问权限要求您对存储桶禁用“屏蔽公共访问权限”设置。有关更多信息以及如何执行此操作，请参阅设置访问网站的权限。要了解如何出于相同的目的设置更严格的策略，请参阅 AWS 知识中心的如何授予对 Amazon S3 存储桶中某些对象的公共读取访问权限？。

默认情况下，Amazon S3 阻止对您的账户和存储桶的公有访问权限。如果要使用存储桶托管静态网站，您可以使用以下步骤编辑您的屏蔽公共访问权限设置。

> **Warning**
>
> 在完成这些步骤之前，请查看阻止对您的 Amazon S3 存储的公有访问，来确保您了解并接受支持公共访问权限所涉及的风险。当您关闭屏蔽公共访问权限设置以使您的存储桶变为公有时，Internet 上的任何人都可以访问您的存储桶。我们建议您阻止对存储桶的所有公有访问。

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 请选择已配置为静态网站的存储桶的名称。
3. 选择权限。
4. 在屏蔽公共访问权限（存储桶设置）下，请选择编辑。
5. 清除阻止所有公有访问，然后选择保存更改。



Amazon S3 关闭了存储桶的屏蔽公共访问权限设置。要创建公有静态网站，可能还必须为您的账户配置屏蔽公共访问权限设置，然后再添加存储桶策略。如果当前已开启账户的屏蔽公共访问权限设置，您将在屏蔽公共访问权限（存储桶设置）下看到一条备注。

需要加密

您可以要求使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS），如以下示例所示。

要求对写入存储桶的所有对象使用 SSE-KMS

以下示例策略要求写入存储桶的每个对象都通过使用 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）进行加密。如果对象未使用 SSE-KMS 进行加密，则请求将被拒绝。

```
{
```

```
"Version": "2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMS",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition": {
    "Null": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "true"
    }
  }
}]
}
```

要求对写入存储桶的所有对象使用具有特定 AWS KMS key 的 SSE-KMS

如果未使用特定的 KMS 密钥 ID 通过 SSE-KMS 加密任何对象，则以下示例策略会拒绝将此类对象写入存储桶。即使使用每请求标头或存储桶默认加密通过 SSE-KMS 加密对象，但如果尚未使用指定的 KMS 密钥加密对象，则无法将其写入存储桶。确保将本示例中使用的 KMS 密钥 ARN 替换为您自己的 KMS 密钥 ARN。

```
{
"Version": "2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMSWithSpecificKey",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
  "Condition": {
    "ArnNotEqualsIfExists": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "arn:aws:kms:us-
east-2:111122223333:key/01234567-89ab-cdef-0123-456789abcdef"
    }
  }
}]
}
```

使用标准 ACL 管理存储桶

授予多个账户上传对象或设置对象 ACL 以进行公有访问的权限

以下示例策略向多个 AWS 账户授予 `s3:PutObject` 和 `s3:PutObjectAcl` 权限。此外，该示例策略还要求针对这些操作的任何请求都必须包含 `public-read` 标准访问控制列表（ACL）。有关更多信息，请参阅Amazon S3 的策略操作 和Amazon S3 的策略条件键。

> **Warning**
>
> `public-read` 标准 ACL 允许世界上的任何人查看您存储桶中的对象。在授予对 Amazon S3 存储桶的匿名访问权限或禁用屏蔽公共访问权限设置时，请小心谨慎。如果您授予匿名访问权限，那么世界上的任何人都可以访问您的存储桶。我们建议您永远不要授予对 Amazon S3 存储桶的匿名访问权限，除非您明确需要，如使用静态网站托管时。如果您想为静态网站托管启用屏蔽公共访问权限设置，请参阅教程：在 Amazon S3 上配置静态网站。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddPublicReadCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root",
                    "arn:aws:iam::444455556666:root"
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
                    ]
                }
            }
        }
```

```
    ]
}
```

在授予上传对象的跨账户权限的同时，确保存储桶拥有者拥有完全控制权

以下示例说明如何允许另一个 AWS 账户将对象上传到存储桶，同时确保您完全控制上传的对象。此策略向特定 AWS 账户（*111122223333*）授予上传对象的能力，但仅当该账户在上传时包含 bucket-owner-full-control 标准 ACL 时才能上传对象。该策略中的 StringEquals 条件指定 s3:x-amz-acl 条件键以表示标准 ACL 要求。有关更多信息，请参阅 Amazon S3 的策略条件键。

```
{
   "Version":"2012-10-17",
   "Statement":[
     {
       "Sid":"PolicyForAllowUploadWithACL",
       "Effect":"Allow",
       "Principal":{"AWS":"111122223333"},
       "Action":"s3:PutObject",
       "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
       "Condition": {
         "StringEquals": {"s3:x-amz-acl":"bucket-owner-full-control"}
       }
     }
   ]
}
```

使用对象标记管理对象访问权限

允许用户仅读取具有特定标签键和值的对象

以下权限策略限制用户只能读取具有 environment: production 标签键和值的对象。该策略使用 s3:ExistingObjectTag 条件键来指定标签键和值。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Principal":{
            "AWS":"arn:aws:iam::111122223333:role/JohnDoe"
         },
         "Effect":"Allow",
         "Action":[
```

```
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
         "Condition":{
            "StringEquals":{
               "s3:ExistingObjectTag/environment":"production"
            }
         }
      }
   ]
}
```

限制用户可以添加哪些对象标签键

以下示例策略将向用户授予执行 s3:PutObjectTagging 操作的权限，这使用户可以将标签添加到现有对象。条件使用 s3:RequestObjectTagKeys 条件键指定允许的标签键，例如 Owner 或 CreationDate。有关更多信息，请参阅《IAM 用户指南》中的创建测试多个键值的条件。

该策略确保在请求中指定的每个标签键都是授权的标签键。条件中的 ForAnyValue 限定符确保请求中必须至少存在指定的值之一。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
```

```
  ]
}
```

在允许用户添加对象标签时需要特定的标签键和值

以下示例策略将向用户授予执行 s3:PutObjectTagging 操作的权限，这使用户可以将标签添加到现有对象。条件要求用户包含值设置为 *X* 的特定标签键（*Project*）。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
         ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

允许用户仅添加具有特定对象标签键和值的对象

以下示例策略向用户授予执行 s3:PutObject 操作的权限，以便他们可以向存储桶添加对象。但是，Condition 语句限制了上传的对象上允许的标签键和值。在此示例中，用户只可以将具有特定标签键（*Department*）（值设置为 *Finance*）的对象添加到存储桶。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Principal":{
            "AWS":[
                 "arn:aws:iam::111122223333:user/JohnDoe"
         ]
```

```
        },
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        ],
        "Condition": {
            "StringEquals": {
                "s3:RequestObjectTag/Department": "Finance"
            }
        }
    }]
}
```

使用全局条件键管理对象访问权限

全局条件键是带有 aws 前缀的条件键。AWS 服务 可以支持全局条件键或包含服务前缀的服务特定键。您可以使用 JSON 策略的 Condition 元素将请求中的键与您在策略中指定的键值进行比较。

限制只能访问 Amazon S3 服务器访问日志传输

在下面的示例存储桶策略中，aws:SourceArn 全局条件键用于比较资源的 Amazon资源名称（ARN），从而使用在策略中指定的 ARN 发出服务对服务请求。aws:SourceArn 全局条件键用于防止 Amazon S3 服务在服务之间执行事务时被用作混淆代理人。只有 Amazon S3 服务才能将对象添加到 Amazon S3 存储桶。

此示例存储桶策略仅向日志服务主体（logging.s3.amazonaws.com）授予 s3:PutObject 权限。

> **Note**
>
> 在 Amazon S3 基于资源的策略（如存储桶策略）中，NotPrincipal 元素不能与 AWS 服务主体一起使用。相反，我们建议使用 aws:PrincipalServiceName 条件键，如以下策略所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowPutObjectS3ServerAccessLogsPolicy",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-logs/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111111111111"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::EXAMPLE-SOURCE-BUCKET"
                }
            }
        },
        {
            "Sid": "RestrictToS3ServerAccessLogs",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET-logs/*",
            "Condition": {
                "StringNotEqualsIfExists": {
                    "aws:PrincipalServiceName": "logging.s3.amazonaws.com"
                }
            }
        }
    ]
}
```

仅允许访问您的组织

如果您希望要求访问资源的所有 IAM 主体都来自组织中的 AWS 账户（包括 AWS Organizations 管理账户），则可以使用 aws:PrincipalOrgID 全局条件键。

要授予或限制此类访问权限，请在存储桶策略中定义 aws:PrincipalOrgID 条件并将值设置为您的组织 ID。组织 ID 用于控制对存储桶的访问权限。当您使用 aws:PrincipalOrgID 条件时，存储桶策略中的权限也将应用于添加到组织的所有新账户。

以下是一个基于资源的存储桶策略的示例，您可以使用该策略向组织中的特定 IAM 主体授予对存储桶的直接访问权限。通过向您的存储桶策略添加 aws:PrincipalOrgID 全局条件键，主体账户现在必须在您的组织中才能访问资源。即使您在授予访问权限时意外指定了错误的账户，aws:PrincipalOrgID

全局条件键也可以起到额外保护作用。当在策略中使用该全局键时，此策略防止指定组织之外的所有主体访问 S3 存储桶。只有来自所列组织中账户的主体才能获得对资源的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowGetObject",
        "Principal": {
            "AWS": "*"
        },
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
        "Condition": {
            "StringEquals": {
                "aws:PrincipalOrgID": ["o-aa111bb222"]
            }
        }
    }]
}
```

管理基于 HTTP 或 HTTPS 请求的访问权限

仅限 HTTPS 请求访问

如果您想防止潜在攻击者操纵网络流量，则可以使用 HTTPS（TLS）以仅允许加密连接，同时限制 HTTP 请求访问您的存储桶。要确定请求是 HTTP 还是 HTTPS，请在 S3 存储桶策略中使用 aws:SecureTransport 全局条件键。`aws:SecureTransport` 条件键检查请求是不是使用 HTTP 发送的。

如果请求返回 `true`，则该请求是通过 HTTPS 发送的。如果请求返回 `false`，则该请求是通过 HTTP 发送的。然后，您可以根据所需的请求方案允许或拒绝对存储桶的访问。

在以下示例中，存储桶策略明确拒绝 HTTP 请求。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "RestrictToTLSRequestsOnly",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": [
```

```
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ],
        "Condition": {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        },
        "Principal": "*"
    }]
}
```

仅限特定 HTTP 引用站点访问

假设您拥有一个网站，其域名为 *www.example.com* 或 *example.com*，并且带有指向存储在名为 *amzn-s3-demo-bucket* 的存储桶中的照片和视频的链接。默认情况下，所有 Amazon S3 资源都是私有的，因此，只有创建资源的 AWS 账户才能访问它们。

要允许从您的网站对这些对象进行读取访问，您可以添加一个存储桶策略，以允许 `s3:GetObject` 权限并附带一个条件，即 GET 请求必须源自特定的网页。以下策略通过使用 `StringLike` 条件和 `aws:Referer` 条件键来限制请求。

```
{
  "Version":"2012-10-17",
  "Id":"HTTP referer policy example",
  "Statement":[
    {
      "Sid":"Allow only GET requests originating from www.example.com and
 example.com.",
      "Effect":"Allow",
      "Principal":"*",
      "Action":["s3:GetObject","s3:GetObjectVersion"],
      "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition":{
        "StringLike":{"aws:Referer":["http://www.example.com/*","http://example.com/
*"]}
      }
    }
  ]
}
```

确保您使用的浏览器在请求中包含 HTTP `referer` 标头。

> **Warning**
>
> 我们建议您谨慎使用 `aws:Referer` 条件键。包含公共已知的 HTTP 引用站点标头值是非常危险的。未经授权方可能会使用修改的浏览器或自定义浏览器提供他们选择的任何 `aws:Referer` 值。因此，请勿使用 `aws:Referer` 防止未经授权的各方直接进行 AWS 请求。
>
> 提供 `aws:Referer` 条件键只是为了允许客户保护其数字内容（如存储在 Amazon S3 中的内容），以免在未经授权的第三方站点上引用。有关更多信息，请参阅 IAM 用户指南中的 aws:Referer。

管理用户对特定文件夹的访问权限

授予用户对特定文件夹的访问权限

假设您正在尝试授予用户对特定文件夹的访问权限。如果 IAM 用户和 S3 存储桶属于同一个 AWS 账户，则您可以使用 IAM 策略向用户授予对特定存储桶文件夹的访问权限。使用这种方法，您无需更新存储桶策略即可授予访问权限。您可以将 IAM 策略添加到多个用户可以切换到的 IAM 角色。

如果 IAM 身份和 S3 存储桶属于不同的 AWS 账户，则您必须在 IAM 策略和存储桶策略中授予跨账户访问权限。有关授予跨账户访问权限的信息，请参阅存储桶拥有者授予跨账户存储桶权限。

以下示例存储桶策略仅授予 *JohnDoe* 对他的文件夹（home/*JohnDoe*/）的完全控制台访问权限。通过创建 home 文件夹并向您的用户授予相应的权限，您可以让多个用户共享单个存储桶。该策略由三条 `Allow` 语句组成：

- *AllowRootAndHomeListingOfCompanyBucket*：允许用户（*JohnDoe*）列出 *DOC-EXAMPLE-BUCKET* 存储桶的根级别和 home 文件夹中的对象。此语句还允许用户使用控制台根据前缀 home/ 进行搜索。
- *AllowListingOfUserFolder*：允许用户（*JohnDoe*）列出 home/*JohnDoe*/ 文件夹和任何子文件夹中的所有对象。
- *AllowAllS3ActionsInUserFolder*：允许用户通过授予 `Read`、`Write` 和 `Delete` 权限来执行所有 Amazon S3 操作。权限仅限于存储桶拥有者的主文件夹。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowRootAndHomeListingOfCompanyBucket",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
            "Condition": {
                "StringEquals": {
                    "s3:prefix": ["", "home/", "home/JohnDoe"],
                    "s3:delimiter": ["/"]
                }
            }
        },
        {
            "Sid": "AllowListingOfUserFolder",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:ListBucket"],
            "Effect": "Allow",
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
            "Condition": {
                "StringLike": {
                    "s3:prefix": ["home/JohnDoe/*"]
                }
            }
        },
        {
            "Sid": "AllowAllS3ActionsInUserFolder",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:*"],
            "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/home/JohnDoe/*"]
        }
    ]
```

```
}
```

管理访问日志的访问权限

授予应用程序负载均衡器访问权限以启用访问日志

在为应用程序负载均衡器启用访问日志时，您必须指定负载均衡器将在其中存储日志的 S3 存储桶的名称。存储桶必须具有附加的策略，用于为弹性负载均衡授予写入存储桶的权限。

在以下示例中，存储桶策略授予弹性负载均衡（ELB）将访问日志写入存储桶的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": "arn:aws:iam::elb-account-id:root"
      },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

> **Note**
>
> 请务必将 *elb-account-id* 替换为您的 AWS 区域的弹性负载均衡的 AWS 账户 ID。有关弹性负载均衡区域的列表，请参阅《弹性负载均衡用户指南》中的将策略附加到您的 Amazon S3 存储桶。

如果您的 AWS 区域未出现在支持的弹性负载均衡区域列表中，请使用以下策略，该策略授予对指定日志传输服务的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
       "Principal": {
```

```
          "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
           },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/111122223333/*"
    }
  ]
}
```

然后，确保通过启用您的弹性负载均衡访问日志来配置它们。您可以通过创建测试文件来验证您的存储桶权限。

管理对 Amazon CloudFront OAI 的访问

向 Amazon CloudFront OAI 授予权限

下面的示例存储桶策略授予 CloudFront 来源访问身份（OAI）权限，以获取（读取）S3 存储桶中的所有对象。您可以使用 CloudFront OAI 以允许用户通过 CloudFront 访问存储桶中的对象，但不能直接通过 Amazon S3 访问。有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的使用来源访问身份限制对 Amazon S3 内容的访问。

下面的策略使用 OAI 的 ID 作为策略的 `Principal`。有关使用 S3 存储桶策略向 CloudFront OAI 授予访问权限的更多信息，请参阅《Amazon CloudFront 开发人员指南》中的从来源访问身份（OAI）迁移到来源访问控制（OAC)。

要使用此示例，请执行以下操作：

- 将 *EH1HDMB1FH2TC* 替换为 OAI 的 ID。要查找 OAI 的 ID，请参阅 CloudFront 控制台中的来源访问身份页面，或者使用 CloudFront API 中的 ListCloudFrontOriginAccessIdentities。
- 将 *amzn-s3-demo-bucket* 替换为您的存储桶的名称。

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
```

```
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

管理 Amazon S3 Storage Lens 存储统计管理工具的访问权限

授予 Amazon S3 Storage Lens 存储统计管理工具的权限

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（存储桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、存储桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 存储桶。

S3 Storage Lens 存储统计管理工具可以将聚合的存储使用情况指标导出到 Amazon S3 存储桶中，以便进一步分析。S3 Storage Lens 存储统计管理工具放置其指标导出的存储桶称为目标存储桶。在设置 S3 Storage Lens 存储统计管理工具指标导出时，您必须为目标存储桶制定存储桶策略。有关更多信息，请参阅 使用 Amazon S3 Storage Lens 存储统计管理工具评估您的存储活动和使用情况。

以下示例存储桶策略授予 Amazon S3 向目标存储桶写入对象（PUT 请求）的权限。在设置 S3 Storage Lens 存储统计管理工具指标导出时，您可以对目标存储桶使用类似这样的存储桶策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3StorageLensExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "storage-lens.s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/destination-prefix/
StorageLens/111122223333/*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": "arn:aws:s3:region-code:111122223333:storage-
lens/storage-lens-dashboard-configuration-id"
                }
            }
        }
    ]
}
```

在设置 S3 Storage Lens 存储统计管理工具组织级别的指标导出时，对之前存储桶策略的 Resource 语句进行以下修改。

```
"Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/destination-prefix/
StorageLens/your-organization-id/*",
```

管理 S3 清单、S3 分析和 S3 清单报告的权限

向 S3 清单和 S3 分析功能授予权限

S3 清单在存储桶中创建对象列表，而 S3 分析存储类分析导出功能创建分析中使用的数据的输出文件。由清单列出其对象的存储桶称为源存储桶。清单文件或分析导出文件将写入到的存储桶称为目标存储桶。在设置清单或分析导出时，必须为目标存储桶创建存储桶策略。有关更多信息，请参阅Amazon S3 清单 和Amazon S3 分析 – 存储类分析。

以下示例存储桶策略向 Amazon S3 授予将源存储桶的账户中的对象写入（PUT 请求）到目标存储桶的权限。您在设置 S3 清单和 S3 分析导出功能时，将对目标存储桶使用像这样的存储桶策略。

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
            "Sid": "InventoryAndAnalyticsExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
```

```
                "arn:aws:s3:::DOC-EXAMPLE-DESTINATION-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

控制 S3 清单报告配置的创建

Amazon S3 清单创建 S3 存储桶中对象的列表，并为每个对象创建元数据。`s3:PutInventoryConfiguration` 权限允许用户创建清单配置（其中包含默认情况下提供的所有可用对象元数据字段）并指定存储清单的目标存储桶。对目标存储桶中的对象具有读取权限的用户可以访问清单报告中提供的所有对象元数据字段。有关 S3 清单中提供的元数据字段的更多信息，请参阅 Amazon S3 清单列表。

要限制用户配置 S3 清单报告，请取消该用户的 `s3:PutInventoryConfiguration` 权限。

S3 清单报告配置中的某些对象元数据字段是可选的，这意味着它们在默认情况下是可用的，但是当您向用户授予 `s3:PutInventoryConfiguration` 权限时，它们可能会受到限制。您可以使用 `s3:InventoryAccessibleOptionalFields` 条件键控制用户能否在报告中包含这些可选元数据字段。有关 S3 清单中可用的可选元数据字段的列表，请参阅《Amazon Simple Storage Service API Reference》中的 OptionalFields。

要向用户授予创建带有特定可选元数据字段的清单配置的权限，请使用 `s3:InventoryAccessibleOptionalFields` 条件键来完善存储桶策略中的条件。

以下策略示例向用户 (*Ana*) 授予有条件地创建清单配置的权限。该策略中的 `ForAllValues:StringEquals` 条件使用 `s3:InventoryAccessibleOptionalFields` 条件键来指定两个允许使用的可选元数据字段，即 `Size` 和 `StorageClass`。因此，在 *Ana* 创建清单配置时，她只可以包含可选元数据字段 `Size` 和 `StorageClass`。

```
{
 "Id": "InventoryConfigPolicy",
```

```
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreationConditionally",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action":
    "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAllValues:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "Size",
        "StorageClass"
        ]
      }
    }
   }
 ]
}
```

要限制用户配置包含特定可选元数据字段的 S3 清单报告，请向源存储桶的存储桶策略中添加明确的 Deny 声明。以下存储桶策略示例拒绝用户 *Ana* 在源存储桶 *DOC-EXAMPLE-SOURCE-BUCKET* 中创建包含可选 ObjectAccessControlList 或 ObjectOwner 元数据字段的清单配置。用户 *Ana* 仍然可以使用其他可选元数据字段创建清单配置。

```
{
 "Id": "InventoryConfigSomeFields",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreation",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",

  },
  {
```

```
    "Sid": "DenyCertainInventoryFieldCreation",
    "Effect": "Deny",
    "Principal": {
     "AWS": "arn:aws:iam::111122223333:user/Ana"
    },
    "Action": "s3:PutInventoryConfiguration",
    "Resource":
      "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
    "Condition": {
     "ForAnyValue:StringEquals": {
      "s3:InventoryAccessibleOptionalFields": [
         "ObjectOwner",
         "ObjectAccessControlList"
         ]
       }
     }
    }
  ]
}
```

> **Note**
>
> 在存储桶策略中使用 s3:InventoryAccessibleOptionalFields 条件键不会影响基于现有清单配置的清单报告交付。

> **Important**
>
> 我们建议将 ForAllValues 与 Allow 效果一起使用，或将 ForAnyValue 与 Deny 效果一起使用，如前面的示例所示。
> 请勿将 ForAllValues 与 Deny 效果一起使用，或将 ForAnyValue 与 Allow 效果一起使用，因为这些组合可能过于严格，并且会阻止删除清单配置。
> 要了解有关 ForAllValues 和 ForAnyValue 条件集运算符的更多信息，请参阅《IAM 用户指南》中的多值上下文键。

需要 MFA

Amazon S3 支持受 MFA 保护的 API 访问，这是一项可在访问您的 Amazon S3 资源时强制进行多重身份验证（MFA）的特征。多重验证提供了额外的安全级别，可以应用于您的 AWS 环境。MFA 是一项

安全特征，要求用户通过提供有效 MFA 代码来证明实际拥有 MFA 设备。有关更多信息，请参阅 AWS 多重身份验证。您可以要求对访问 Amazon S3 资源的任何请求使用 MFA。

要强制实施 MFA 要求，请在存储桶策略中使用 aws:MultiFactorAuthAge 条件键。IAM 用户可以使用由 AWS Security Token Service（AWS STS）发布的临时凭证访问 Amazon S3 资源。您可以在 AWS STS 请求时提供 MFA 代码。

当 Amazon S3 收到带多重身份验证的请求时，aws:MultiFactorAuthAge 条件键将提供一个数值，指示临时凭证是在多久以前创建的（以秒为单位）。如果请求中提供的临时凭证不是使用 MFA 设备创建的，则此键值为空（缺失）。您可以在存储桶策略中添加一个条件来检查此值，如下面的示例所示。

如果请求未使用 MFA 进行身份验证，此示例策略将拒绝对 *amzn-s3-demo-bucket* 存储桶中的 */taxdocuments* 文件夹执行任何 Amazon S3 操作。要了解有关 MFA 的更多信息，请参阅 IAM 用户指南中的在 AWS 中使用多重身份验证 (MFA)。

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true }}
      }
    ]
 }
```

如果 aws:MultiFactorAuthAge 条件键值为空，即指示请求中的临时安全凭证是在没有 MFA 设备的情况下创建的，则 Condition 块中的 Null 条件的计算结果为 true。

下面的存储桶策略是上述存储桶策略的扩展。以下策略包含两个策略语句。一个语句允许将存储桶（*amzn-s3-demo-bucket*）的 s3:GetObject 权限授予所有人。另一个语句通过要求 MFA，进一步限制对 *amzn-s3-demo-bucket/taxdocuments* 文件夹的访问。

```
{
    "Version": "2012-10-17",
```

```
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }
      },
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": "*",
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      }
    ]
}
```

您可以选择使用数值条件来限制 aws:MultiFactorAuthAge 键的有效期。您使用 aws:MultiFactorAuthAge 键指定的期限独立于对请求进行身份验证时使用的临时安全凭证的生存期。

例如，除了要求 MFA 身份验证外，下面的存储桶策略还会查看临时会话是在多久以前创建的。如果 aws:MultiFactorAuthAge 键值指示临时会话是在一个小时 (3600 秒) 之前创建的，则策略将拒绝任何操作。

```
{
    "Version": "2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": {"Null": {"aws:MultiFactorAuthAge": true }}
      },
      {
        "Sid": "",
        "Effect": "Deny",
```

```
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": {"NumericGreaterThan": {"aws:MultiFactorAuthAge": 3600 }}
      },
      {
        "Sid": "",
        "Effect": "Allow",
        "Principal": "*",
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      }
    ]
}
```

防止用户删除对象

默认状态下，用户没有权限。但在创建策略时，您可能无意间向用户授予了您并不打算授予的权限。为避免这些权限漏洞，可通过添加显式拒绝编写更严格的访问策略。

要显式阻止用户或账户删除对象，您必须在存储桶策略中添加以下操作：s3:DeleteObject、s3:DeleteObjectVersion 和 s3:PutLifecycleConfiguration 权限。所有三个操作均为必需，因为您可通过显式调用 DELETE Object API 操作或配置其生命周期来删除对象（请参阅管理存储生命周期），以便 Amazon S3 能够在对象生命周期已过时将它们移除。

在以下策略示例中，您显式对用户 *MaryMajor* 拒绝 DELETE Object 权限。显式 Deny 语句始终取代授予的其它任何权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/MaryMajor"
      },
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetBucketAcl"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::amzn-s3-demo-bucket1",
    "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    },
    {
      "Sid": "statement2",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/MaryMajor"
      },
      "Action": [
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutLifecycleConfiguration"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
     "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

使用条件键的存储桶策略示例

您可以使用访问策略语言在授予权限时指定条件。您可以使用可选 Condition 元素或 Condition 块来指定策略生效的条件。

有关使用 Amazon S3 条件键进行对象和存储桶操作的策略，请参阅以下示例。有关条件键的更多信息，请参阅 Amazon S3 的策略条件键。有关您可以在策略中指定的 Amazon S3 操作、条件键和资源的完整列表，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

示例 — 针对对象操作的 Amazon S3 条件键

本节提供示例，说明如何将特定于 Amazon S3 的条件键用于对象操作。有关您可以在策略中指定的 Amazon S3 操作、条件键和资源的完整列表，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

几个示例策略展示如何将条件键与 PUT Object 操作结合使用。PUT Object 操作允许特定于访问控制列表 (ACL) 的标头，可用于授予基于 ACL 的权限。通过使用这些键，存储桶拥有者可设置条件，要求用户上传对象时需具有特定访问权限。您还可以通过 PutObjectAcl 操作授予基于 ACL 的权限。有关更

多信息，请参阅 Amazon S3 Amazon Simple Storage Service API 参考中的 PutObjectAcl。有关 ACL 的更多信息，请参阅 访问控制列表 (ACL) 概述。

主题

示例 1：授予要求使用服务器端加密存储对象的 s3:PutObject 权限

假设账户 A 拥有一个存储桶。账户管理员想要授予账户 A 中的用户 Jane 上传对象的权限，条件是 Jane 始终请求服务器端加密，使 Amazon S3 保存加密的对象。账户 A 管理员可使用所示的 `s3:x-amz-server-side-encryption` 条件键来完成。`Condition` 块中的键值对指定 `s3:x-amz-server-side-encryption` 键。

```
"Condition": {
     "StringNotEquals": {
         "s3:x-amz-server-side-encryption": "AES256"
     }}
```

当使用 AWS CLI 测试此权限时，必须使用 `--server-side-encryption` 参数添加所需的参数。

```
aws s3api put-object --bucket example1bucket --key HappyFace.jpg --body c:
\HappyFace.jpg --server-side-encryption "AES256" --profile AccountBadmin
```

示例 2：授予从限定复制源复制对象的 s3:PutObject 权限

在 PUT 对象请求中，如果指定了源对象，则为一个复制操作（请参阅 PUT 对象 - 复制）。因此，存储桶拥有者可以为用户授予权限以复制具有源限制的对象，例如：

- 允许仅从 `sourcebucket` 存储桶复制对象。
- 允许从源存储桶复制对象，并仅复制键名称前缀开头为 public/ 的对象（例如 sourcebucket/public/*）。
- 仅允许从源存储桶复制特定对象（例如，sourcebucket/example.jpg）。

以下存储桶策略为用户（Dave）授予 s3:PutObject 权限。该权限允许用户仅复制满足以下条件的对象：请求包含 s3:x-amz-copy-source 标头，并且标头值指定 /awsexamplebucket1/public/* 键名称前缀。

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "cross-account permission to user in your own account",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*"
        },
        {
            "Sid": "Deny your user permission to upload object if copy source is not /
bucket/folder",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringNotLike": {
                    "s3:x-amz-copy-source": "awsexamplebucket1/public/*"
                }
            }
        }
    ]
}
```

使用 AWS CLI 测试策略

可使用 AWS CLI copy-object 命令测试此权限。可通过添加 --copy-source 参数指定源；键名称前缀必须与策略中允许的前缀相匹配。您需要使用 --profile 参数为用户 Dave 提供凭证。有关设置 AWS CLI 的更多信息，请参阅 使用 AWS CLI 进行 Amazon S3 开发。

```
aws s3api copy-object --bucket awsexamplebucket1 --key HappyFace.jpg
--copy-source examplebucket/public/PublicHappyFace1.jpg --profile AccountADave
```

授予仅复制特定对象的权限

上述策略使用 StringNotLike 条件。要授予仅复制特定对象的权限，您必须将条件从 StringNotLike 更改为 StringNotEquals，然后指定所示的对象键。

```
"Condition": {
       "StringNotEquals": {
           "s3:x-amz-copy-source": "awsexamplebucket1/public/PublicHappyFace1.jpg"
       }
}
```

示例 3：授予对特定对象版本的访问权限

假设账户 A 拥有启用版本控制的存储桶。该存储桶具有 HappyFace.jpg 对象的多个版本。账户管理员现在想要授予用户 Dave 仅获得特定对象版本的权限。账户管理员可通过有条件地授予 Dave 下面所示的 s3:GetObjectVersion 权限来实现这一点。Condition 块中的键值对指定 s3:VersionId 条件键。在这种情况下，Dave 需要知道确切的对象版本 ID 才能检索该对象。

有关更多信息，请参阅 Amazon Simple Storage Service API 参考中的 GetObject。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg"
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::examplebucketversionenabled/HappyFace.jpg",
            "Condition": {
                "StringNotEquals": {
                    "s3:VersionId": "AaaHbAQitwiL_h47_44lRO2DDfLlBO5e"
```

```
                }
            }
        }
    ]
}
```

使用 AWS CLI 测试策略

可使用 AWS CLI `get-object` 命令以及标识特定对象版本的 `--version-id` 参数来测试这些权限。此命令会检索该对象，并将其保存到 `OutputFile.jpg` 文件。

```
aws s3api get-object --bucket examplebucketversionenabled --key HappyFace.jpg
 OutputFile.jpg --version-id AaaHbAQitwiL_h47_44lRO2DDfLlBO5e --profile AccountADave
```

示例 4：基于对象标签授予权限

有关如何将对象标签条件键与 Amazon S3 操作结合使用的示例，请参阅标签和访问控制策略。

示例 5：限制存储桶拥有者的 AWS 账户 ID 的访问

您可以使用 `aws:ResourceAccount` 或 `s3:ResourceAccount` 键编写 IAM 或虚拟私有云（VPC）端点策略，以限制用户、角色或应用程序对特定 AWS 账户 ID 所拥有的 Amazon S3 存储桶的访问权限。您可以使用此条件键来限制 VPC 内的客户端访问非您所有的存储桶。

但请注意，有些 AWS 服务依赖于访问 AWS 托管式存储桶。因此，在 IAM 策略中使用 `aws:ResourceAccount` 或 `s3:ResourceAccount` 键也可能会影响对这些资源的访问。

有关更多信息和示例，请参阅以下资源：

- 《AWS PrivateLink 指南》中的限制对指定 AWS 账户中存储桶的访问
- 《Amazon ECR 指南》中的限制对 Amazon ECR 使用的存储桶的访问
- AWS 指南中的为 AWS Systems Manager 托管式 Amazon S3 存储桶提供对 Systems Manager 的所需访问权限
- AWS 存储博客 中的将访问权限限于特定 AWS 账户拥有的 Amazon S3 存储桶

示例 6：要求最低 TLS 版本

您可以使用 s3:TlsVersion 条件键来编写 IAM、Virtual Private Cloud 端点（VPCE）或存储桶策略，以根据客户端使用的 TLS 版本限制用户或应用程序对 Amazon S3 存储桶的访问。您可以使用此条件键来编写要求最低 TLS 版本的策略。

Example

此示例存储桶策略拒绝 TLS 版本低于 1.2 的客户端（例如 1.1 或 1.0）发出的 PutObject 请求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ],
            "Condition": {
                "NumericLessThan": {
                    "s3:TlsVersion": 1.2
                }
            }
        }
    ]
}
```

Example

此示例存储桶策略允许 TLS 版本高于 1.1 的客户端（例如 1.2、1.3 或更高版本）发出的 PutObject 请求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ],
```

```
            "Condition": {
                "NumericGreaterThan": {
                    "s3:TlsVersion": 1.1
                }
            }
        }
    ]
}
```

示例 — 针对存储桶操作的 Amazon S3 条件键

本节提供示例策略，说明如何将特定于 Amazon S3 的条件键用于存储桶操作。

主题

- 示例 1：授予 s3:GetObject 权限，指定了 IP 地址的条件
- 示例 2：获取存储桶中具有特定前缀的对象列表
- 示例 3：设置最大键数

示例 1：授予 s3:GetObject 权限，指定了 IP 地址的条件

如果请求源自特定 IP 地址范围（192.0.2.*），则您可以为已经过身份验证的用户授予使用 `s3:GetObject` 操作的权限，除非此 IP 地址为 192.0.2.188。在条件块中，`IpAddress` 和 `NotIpAddress` 为条件，每个条件均提供了一个键值对用于评估。此示例中的两个键-值对均使用 `aws:SourceIp` AWS 范围内的键。

> **Note**
>
> 请注意，在条件中指定的 `IPAddress` 和 `NotIpAddress` 键值使用 RFC 4632 中描述的 CIDR 表示法。有关更多信息，请参阅 http://www.rfc-editor.org/rfc/rfc4632.txt。

```
{
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": "*",
```

```
            "Action":"s3:GetObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
                    "aws:SourceIp": "192.0.2.188/32"
                }
            }
        }
    ]
}
```

您还可以在 Amazon S3 策略中使用其他 AWS 范围的条件键。例如，您可以在适用于 VPC 端点的存储桶策略中指定 `aws:SourceVpce` 和 `aws:SourceVpc` 条件键。有关特定示例，请参阅[使用存储桶策略控制从 VPC 端点的访问]。

> **Note**
>
> 对于某些 AWS 全局条件键，仅支持某些资源类型。因此，请检查 Amazon S3 是否支持您要使用的全局条件键和资源类型，或者是否需要改用 Amazon S3 特定的条件键。有关 Amazon S3 的受支持资源类型和条件键的完整列表，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

示例 2：获取存储桶中具有特定前缀的对象列表

您可以使用 `s3:prefix` 条件键将 GET Bucket (ListObjects) API 的响应限制为具有特定前缀的键名。如果您是存储桶拥有者，您可限定用户仅列出存储桶中特定前缀的内容。如果存储桶中的对象按键名前缀组织，此条件键非常有用。Amazon S3 控制台使用键名前缀来显示文件夹概念。只有控制台支持文件夹的概念；Amazon S3 API 仅支持存储桶和对象。有关使用前缀和分隔符筛选访问权限的更多信息，请参阅 使用用户策略控制对存储桶的访问。

例如，如果您有键名为 `public/object1.jpg` 和 `public/object2.jpg` 的两个对象，则该控制台会在 `public` 文件夹下显示这些对象。在 Amazon S3 API 中，这些是带有前缀的对象，而不是文件夹中的对象。但是，在 Amazon S3 API 中，如果使用这些前缀组织对象键，则可授予 `s3:ListBucket` 权限，`s3:prefix` 条件为允许用户获得具有这些特定前缀的键名的列表。

在该示例中，存储桶拥有者和用户所属的父账户相同。因此存储桶拥有者可使用存储桶策略或用户策略。有关可与 GET Bucket (ListObjects) API 一起使用的其他条件键的更多信息，请参阅 ListObjects。

用户策略

以下用户策略授予 s3:ListBucket 权限（请参阅 GET Bucket (List Objects)），条件为要求用户在请求中指定值为 prefix 的 projects。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:ListBucket",
         "Resource":"arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
    ]
}
```

此条件将用户限定于列出具有 projects 前缀的对象键。添加的显式拒绝将拒绝用户列出具有其他任何前缀的键，无论该用户可能具有其他什么权限。例如，通过更新先前用户策略或通过存储桶策略，该用户有可能获得列出没有任何限制的对象键的权限。由于显式拒绝始终优先于其他任何权限，因此列出非 projects 前缀的键的用户请求会被拒绝。

存储桶策略

如果将 Principal 元素添加到上述的用户策略，标识用户，则现在您拥有了所示的存储桶策略。

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action":  "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::awsexamplebucket1",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
       }
    ]
}
```

使用 AWS CLI 测试策略

可使用以下 `list-object` AWS CLI 命令测试此策略。在该命令中，使用 `--profile` 参数提供用户凭证。有关设置和使用 AWS CLI 的更多信息，请参阅 使用 AWS CLI 进行 Amazon S3 开发。

```
aws s3api list-objects --bucket awsexamplebucket1 --prefix examplefolder --profile
 AccountADave
```

如果该存储桶启用了版本控制，要列出该存储桶中的对象，必须在上述策略中授予 `s3:ListBucketVersions` 权限，而不是 `s3:ListBucket` 权限。此权限还支持 `s3:prefix` 条件键。

示例 3：设置最大键数

您可以使用 `s3:max-keys` 条件键设置请求者在 GET Bucket (ListObjects) 或 ListObjectVersions 请求中可以返回的最大键数。默认情况下，API 返回最多 1000 个键。有关可与 `s3:max-keys` 一起使用的数字条件运算符的列表和相关示例，请参阅 IAM 用户指南中的数字条件运算符。

Amazon S3 基于身份的策略

默认情况下，用户和角色没有创建或修改 Amazon S3 资源的权限。他们也无法使用 AWS Management Console、AWS Command Line Interface（AWS CLI）或 AWS API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅 IAM 用户指南中的 创建 IAM 策略。

有关 Amazon S3 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

主题

- 策略最佳实践
- Amazon S3 基于身份的策略示例
- 使用用户策略控制对存储桶的访问

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon S3 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- AWS 托管策略及转向最低权限许可入门 – 要开始向用户和工作负载授予权限，请使用 AWS 托管策略来为许多常见使用场景授予权限。您可以在 AWS 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 AWS 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略或工作职能的 AWS 托管策略。

- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定（AWS 服务例如 AWS CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证（MFA）– 如果您所处的场景要求您的 AWS 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实操。

Amazon S3 基于身份的策略示例

本节介绍用于控制对 Amazon S3 的访问权限的几个基于 AWS Identity and Access Management（IAM）身份的策略示例。有关存储桶策略（基于资源的策略）的示例，请参阅 Amazon S3 的存储桶策略。有关 IAM 策略语言的信息，请参阅Amazon S3 中的策略和权限。

如果以编程方式使用以下示例策略，它们将正常工作。不过，要在 Amazon S3 控制台中使用这些策略，您必须授予控制台所需的额外权限。有关使用策略（例如与 Amazon S3 控制台一起使用的策略）的信息，请参阅 使用用户策略控制对存储桶的访问。

主题

- 允许 IAM 用户访问某个存储桶
- 允许每个 IAM 用户访问存储桶中的文件夹
- 允许组在 Amazon S3 中拥有共享的文件夹
- 允许所有用户读取存储桶的某个部分中的对象
- 允许合作伙伴将文件放置到存储桶的特定部分中
- 限制对特定 AWS 账户中 Amazon S3 存储桶的访问
- 限制企业单位内对 Amazon S3 存储桶的访问

- 限制对企业内 Amazon S3 存储桶的访问
- 为 AWS 账户授予检索 PublicAccessBlock 配置的权限
- 将存储桶的创建限制在一个区域

允许 IAM 用户访问某个存储桶

在本示例中，您需要授予您的 AWS 账户中的 IAM 用户访问一个存储桶 *amzn-s3-demo-bucket1* 的权限，以便该用户能够添加、更新和删除对象。

除了授予该用户 s3:PutObject、s3:GetObject 和 s3:DeleteObject 权限外，此策略还授予 s3:ListAllMyBuckets、s3:GetBucketLocation 和 s3:ListBucket 权限。这些是控制台所需的其他权限。此外，s3:PutObjectAcl 和 s3:GetObjectAcl 操作需要能够在控制台中复制、剪切和粘贴对象。有关向用户授予权限并使用控制台测试这些权限的示例演练，请参阅 使用用户策略控制对存储桶的访问。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action": "s3:ListAllMyBuckets",
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":["s3:ListBucket","s3:GetBucketLocation"],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:GetObject",
            "s3:GetObjectAcl",
            "s3:DeleteObject"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*"
      }
   ]
}
```

允许每个 IAM 用户访问存储桶中的文件夹

在本示例中，您需要两个 IAM 用户（Mary 和 Carlos）具有访问存储桶 *amzn-s3-demo-bucket1* 的权限，以便他们可以添加、更新和删除对象。但是，您想要限制每个用户对存储桶中单个前缀（文件夹）的访问权限。您可以使用与其用户名匹配的名称创建文件夹。

```
amzn-s3-demo-bucket1
   Mary/
   Carlos/
```

要授予每个用户仅可以访问其文件夹的权限，您可以为每个用户编写策略，然后分别附加它。例如，您可以将以下策略附加到用户 Mary，以允许她对 *amzn-s3-demo-bucket1*/*Mary* 文件夹拥有特定的 Amazon S3 权限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/Mary/*"
      }
   ]
}
```

然后，您可以将类似策略附加到用户 Carlos，同时在 `Resource` 值中指定文件夹 *Carlos*。

您可以编写一个使用策略变量的策略，然后将该策略附加到一个组，而不是将策略附加到单个用户。首先，您必须创建一个组，并将 Mary 和 Carlos 添加到该组中。以下示例策略允许在 *amzn-s3-demo-bucket1*/${aws:username} 文件夹中具有一组 Amazon S3 权限。评估策略后，策略变量 ${aws:username} 将替换为请求者的用户名。例如，如果 Mary 发送了一个请求以放置对象，只有当 Mary 将对象上传到 *amzn-s3-demo-bucket1*/Mary 文件夹后，才允许该操作。

```
{
   "Version":"2012-10-17",
```

```
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/${aws:username}/*"
      }
   ]
}
```

> **Note**
>
> 当使用策略变量时，您必须在策略中明确指定版本 2012-10-17。IAM 策略语言的默认版本 2008-10-17 不支持策略变量。

如果需要在 Amazon S3 控制台上测试之前的策略，控制台需要其他权限，如以下策略所示。有关控制台如何使用这些权限的信息，请参阅 使用用户策略控制对存储桶的访问。

```
{
 "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": [
       "s3:ListAllMyBuckets",
       "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "AllowRootLevelListingOfTheBucket",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1",
      "Condition":{
```

```
            "StringEquals":{
                    "s3:prefix":[""], "s3:delimiter":["/"]
                           }
                }
    },
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1",
      "Condition":{  "StringLike":{"s3:prefix":["${aws:username}/*"] }
       }
    },
      {
     "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/${aws:username}/*"
      }
  ]
}
```

> **Note**
>
> 在 2012-10-17 版本的策略中，策略变量以 $ 开始。如果您的对象键（对象名称）包括 $，则语法中的此更改可能会产生冲突。
> 为避免此冲突，请通过使用 ${$} 指定 $ 字符。例如，要在策略中包括对象键 my$file，请将其指定为 my${$}file。

尽管 IAM 用户名称是友好、用户可读的标识符，但是它们无需全局唯一。例如，如果用户 Carlos 离开了企业，而另一个 Carlos 加入进来，则新 Carlos 可以访问原 Carlos 的信息。

您可以基于 IAM 用户 ID 创建文件夹，而不是使用用户名。每个 IAM 用户 ID 都是唯一的。在这种情况下，您必须修改之前的策略，以使用 ${aws:userid} 策略变量。有关用户标识符的更多信息，请参阅 IAM 用户指南中的 IAM 标识符。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/home/${aws:userid}/*"
      }
   ]
}
```

允许非 IAM 用户（移动应用程序用户）访问存储桶中的文件夹

假设您要开发一个移动应用程序，一个将用户数据存储在 S3 存储桶中的游戏。对于每个应用用户，您都需要在您的存储桶中创建一个文件夹。您还需要限制每个用户对其自己的文件夹的访问权限。但是，在有人下载您的应用程序并开始玩此游戏之前，您不能创建文件夹，因为您没有用户 ID。

这种情况下，您可以要求用户使用公共身份提供商 (如 Login with Amazon、Facebook、或 Google) 登录到您的应用程序。在用户通过某个提供商登录到您的应用程序后，他们具有一个用户 ID，可用于在运行时创建用户特定的文件夹。

然后，您可以使用 AWS Security Token Service 中的 Web 联合身份验证将来自身份提供商的信息与您的应用相集成，为每个用户获取临时安全凭证。然后，您可以创建 IAM 策略，以便允许该应用访问存储桶和执行操作，如创建用户特定文件夹和上传数据。有关 Web 身份联合验证的更多信息，请参阅《IAM 用户指南》中的关于 Web 身份联合验证。

允许组在 Amazon S3 中拥有共享的文件夹

将以下策略附加到组将授予该组中的每个人访问 Amazon S3 中的以下文件夹的权限：*amzn-s3-demo-bucket1*/share/marketing。组成员仅允许访问策略中显示的特定 Amazon S3 权限，仅适用于指定文件夹中的对象。

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/share/marketing/*"
      }
   ]
}
```

允许所有用户读取存储桶的某个部分中的对象

在该示例中，您创建一个名为 *AllUsers* 的组，其中包含 AWS 账户拥有的所有 IAM 用户。然后，您将附加向该组提供对 GetObject 和 GetObjectVersion 的访问权限的策略，但仅适用于 *amzn-s3-demo-bucket1*/readonly 文件夹中的对象。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/readonly/*"
      }
   ]
}
```

允许合作伙伴将文件放置到存储桶的特定部分中

在本示例中，您将创建代表合作伙伴公司的名为 *AnyCompany* 的组。您将为需要访问权限的合作伙伴公司中的特定人员或应用程序创建 IAM 用户，然后将该用户放入组中。

然后，您附加策略，此策略向组提供对存储桶中以下文件夹的 PutObject 访问权限：

amzn-s3-demo-bucket1/uploads/anycompany

您需要阻止 *AnyCompany* 组对存储桶执行任何其他操作，因此添加了一条语句，除了对 AWS 账户中的任何 Amazon S3 资源执行 PutObject 外，该语句显式拒绝执行任何其他 Amazon S3 操作的权限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/uploads/anycompany/*"
      },
      {
         "Effect":"Deny",
         "Action":"s3:*",
         "NotResource":"arn:aws:s3:::amzn-s3-demo-bucket1/uploads/anycompany/*"
      }
   ]
}
```

限制对特定 AWS 账户中 Amazon S3 存储桶的访问

如果您想确保您的 Amazon S3 主体只能访问受信任 AWS 账户中的资源，您可以限制访问权限。例如，基于身份的 IAM 策略使用 Deny 影响来阻止对 Amazon S3 操作的访问，除非正在访问的 Amazon S3 资源在账户 *222222222222* 中。为了防止 AWS 账户中的 IAM 主体访问账户外的 Amazon S3 对象，请附加以下 IAM 策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyS3AccessOutsideMyBoundary",
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "222222222222"
          ]
```

```
        }
      }
    }
  ]
}
```

> **Note**
>
> 此策略不会取代现有的 IAM 访问控制，因为它不授予任何访问权限。相反，无论通过其他 IAM 策略授予什么权限，此策略都可以作为其他 IAM 权限的额外防护机制。

请确保将策略中的账户 ID *222222222222* 替换为您自己的 AWS 账户。要在仍保持此限制的同时将策略应用于多个账户，请将账户 ID 替换为 `aws:PrincipalAccount` 条件键。该条件要求主体和资源必须在同一账户中。

限制企业单位内对 Amazon S3 存储桶的访问

如果在 AWS Organizations 中设置了企业单位（OU），则可能需要将 Amazon S3 存储桶访问限制为企业的特定部分。在本示例中，我们将使用 `aws:ResourceOrgPaths` 键以限制 Amazon S3 存储桶对企业中 OU 的访问。在本例中，OU ID 为 *ou-acroot-exampleou*。确保用自己的 OU ID 替换自己的策略中的此值。

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "AllowS3AccessOutsideMyBoundary",
       "Effect": "Allow",
       "Action": [
         "s3:*"
       ],
       "Resource": "*",
       "Condition": {
         "ForAllValues:StringNotLike": {
           "aws:ResourceOrgPaths": [
             "o-acorg/r-acroot/ou-acroot-exampleou/"
           ]
         }
       }
     }
```

```
    ]
}
```

> **Note**
>
> 该策略不授予任何访问权限。相反，此策略充当其他 IAM 权限的后盾，阻止主体访问 OU 定义边界之外的 Amazon S3 对象。

该策略拒绝对 Amazon S3 操作的访问，除非正在访问的 Amazon S3 对象位于您企业的 *ou-acroot-exampleou* OU 中。IAM 策略条件要求 `aws:ResourceOrgPaths`（多值条件键）包含列出的任何 OU 路径。该策略使用 `ForAllValues:StringNotLike` 运算符将 `aws:ResourceOrgPaths` 的值与列出的 OU 进行比较，而不进行区分大小写的匹配。

限制对企业内 Amazon S3 存储桶的访问

要限制对组织内 Amazon S3 对象的访问，请将 IAM 策略附加到组织的根，并将其应用于组织中的所有账户。如果要求您的 IAM 主体遵守此规则，请使用服务控制策略 (SCP)。如果您选择使用 SCP，请确保先全面测试 SCP，然后将策略附加到企业的根。

在以下示例策略中，将拒绝对 Amazon S3 操作的访问，除非正在访问的 Amazon S3 对象与访问该对象的 IAM 主体位于同一企业中：

```
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Sid": "DenyS3AccessOutsideMyBoundary",
       "Effect": "Deny",
       "Action": [
         "s3:*"
       ],
       "Resource": "arn:aws:s3:::*/*",
       "Condition": {
         "StringNotEquals": {
           "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
         }
       }
     }
   ]
}
```

> **Note**
>
> 该策略不授予任何访问权限。相反，此策略充当其他 IAM 权限的后盾，阻止主体访问企业外部的任何 Amazon S3 对象。此策略还适用于在策略生效后创建的 Amazon S3 资源。

本例中的 IAM 策略条件要求 `aws:ResourceOrgID` 和 `aws:PrincipalOrgID` 彼此相等。有了这一要求，提出请求的主体和正在访问的资源必须位于同一企业中。

为 AWS 账户授予检索 PublicAccessBlock 配置的权限

以下基于身份的策略示例向用户授予 `s3:GetAccountPublicAccessBlock` 权限。对于这些权限，请将 `Resource` 值设置为 `"*"`。有关资源 ARN 的更多信息，请参阅Amazon S3 的策略资源。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

将存储桶的创建限制在一个区域

假定 AWS 账户管理员想要授予其用户（Dave）仅在南美洲（圣保罗）区域创建存储桶的权限。账户管理员可附加以下用户策略，授予附带条件的 `s3:CreateBucket` 权限，如下所示。`Condition` 块中的键值对指定 `s3:LocationConstraint` 键，并将 `sa-east-1` 区域作为值。

> **Note**
>
> 在该示例中，存储桶拥有者为其用户之一授予权限，因此可以使用存储桶策略或用户策略。此示例显示了用户策略。

有关 Amazon S3 区域的列表，请参阅《AWS 一般参考》中的区域和端点。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

添加显式拒绝

前面的策略限制用户只能在 sa-east-1 区域中创建存储桶。但是，别的策略可能授予此用户在其他区域中创建存储桶的权限。例如，如果用户属于某个组，该组可能附加了一个策略，以允许该组中的所有用户在另一个区域中创建存储桶。要确保此用户不会获得在其它任何区域创建存储桶的权限，可在上述策略中添加一个显式拒绝语句。

Deny 语句使用 StringNotLike 条件。也即，如果位置约束不是 sa-east-1，则创建存储桶的请求将被拒绝。显式拒绝不允许用户在其它任何区域创建存储桶，无论该用户获得了哪种其它权限。以下策略包含显式拒绝语句。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
```

```
          }
        },
       {
          "Sid":"statement2",
          "Effect":"Deny",
          "Action": "s3:CreateBucket",
          "Resource": "arn:aws:s3:::*",
          "Condition": {
              "StringNotLike": {
                  "s3:LocationConstraint": "sa-east-1"
              }
          }
        }
    ]
}
```

使用 AWS CLI 测试策略

可使用以下 `create-bucket` AWS CLI 命令测试此策略。此示例使用 `bucketconfig.txt` 文件来指定位置约束。记下 Windows 文件路径。您需要更新相应的存储桶名称和路径。必须使用 `--profile` 参数提供用户凭证。有关设置和使用 AWS CLI 的更多信息，请参阅 使用 AWS CLI 进行 Amazon S3 开发。

```
aws s3api create-bucket --bucket examplebucket --profile AccountADave --create-bucket-configuration file://c:/Users/someUser/bucketconfig.txt
```

`bucketconfig.txt` 文件指定一些配置，如下所示。

```
{"LocationConstraint": "sa-east-1"}
```

使用用户策略控制对存储桶的访问

本演练将介绍用户许可在 Amazon S3 中的使用方式。在该示例中，您创建一个具有文件夹的存储桶。然后，在您的 AWS 账户 中创建 AWS Identity and Access Management IAM 用户，并为这些用户授予对 Amazon S3 存储桶以及其中文件夹的增量权限。

主题

- 存储桶和文件夹基础知识
- 演练摘要

存储桶和文件夹基础知识

Amazon S3 数据模型是一种扁平结构：您创建一个存储桶，存储桶存储对象。没有子存储桶或子文件夹层次结构，但您可以模拟文件夹层次结构。Amazon S3 控制台等工具可以显示存储桶中这些逻辑文件夹和子文件夹的视图。

控制台显示名为 companybucket 的存储桶具有三个文件夹（Private、Development 和 Finance）和一个对象（s3-dg.pdf）。控制台使用对象名称 (键) 来创建文件夹和子文件夹的逻辑层次结构。考虑以下示例：

- 在创建 Development 文件夹时，控制台使用 Development/ 键创建一个对象。请注意尾部斜杠 (/) 分隔符。
- 在上传 Projects1.xls 文件夹中名为 Development 的对象时，控制台上传该对象并为其分配 Development/Projects1.xls 键。

  在该键值中，Development 是前缀，/ 是分隔符。Amazon S3 API 在运行时支持前缀和分隔符。例如，您可以使用特定的前缀和分隔符，从存储桶获取所有对象的列表。在控制台上，在打开 Development 文件夹时，控制台列出该文件夹中的对象。在以下示例中，Development 文件夹包含一个对象。

  当控制台列出 Development 存储桶中的 companybucket 文件夹时，它向 Amazon S3 发送一个请求，在请求中指定 Development 前缀和 / 分隔符。控制台的反应方式与计算机文件系统内的文件夹列表的反应方式相似。上一个示例显示，存储桶 companybucket 拥有一个带键值 Development/Projects1.xls 的对象。

控制台使用对象键来推断逻辑层次结构。Amazon S3 没有物理层次结构。Amazon S3 只拥有包含平面文件结构中的对象的存储桶。使用 Amazon S3 API 创建对象时，您可以使用暗指逻辑层次结构的对象键。在创建对象的逻辑层次结构时，您可以管理对单个文件夹的访问，如本演练中所示。

在开始之前，请确保您熟悉根级 存储桶内容概念。假设 companybucket 存储桶具有以下对象：

- Private/privDoc1.txt
- Private/privDoc2.zip
- Development/project1.xls
- Development/project2.xls
- Finance/Tax2011/document1.pdf
- Finance/Tax2011/document2.pdf
- s3-dg.pdf

这些对象键创建一个逻辑层次结构，将 Private、Development 和 Finance 作为根级文件夹并将 s3-dg.pdf 作为根级对象。在 Amazon S3 控制台上选择存储桶名称时，将显示根级项目。控制台将顶级前缀（Private/、Development/ 和 Finance/）显示为根级文件夹。s3-dg.pdf 对象键没有前缀，因此，它显示为根级项目。

演练摘要

在本演练中，您创建一个包含三个文件夹（Private、Development 和 Finance）的存储桶。

您拥有两个用户，Alice 和 Bob。您希望 Alice 仅访问 Development 文件夹，并希望 Bob 仅访问 Finance 文件夹。您希望将 Private 文件夹内容保密。在本演练中，您创建 IAM 用户（该示例使用用户名 Alice 和 Bob）并为其授予所需的权限以管理访问。

IAM 还支持创建用户组以及授予适用于组中所有用户的组级许可。这将帮助您更好地管理权限。在该练习中，Alice 和 Bob 需要具有一些相同的权限。因此，您还创建一个名为 Consultants 的组，然后将 Alice 和 Bob 添加到该组中。您先将一个组策略附加到该组以授予权限。然后，您将策略附加到特定用户以添加用户特定的权限。

> **Note**
>
> 本演练将 companybucket 作为存储桶名称，将 Alice 和 Bob 作为 IAM 用户并将 Consultants 作为组名称。由于 Amazon S3 要求存储桶名称全局唯一，因此，您必须将存储桶名称替换为您创建的名称。

准备演练

在该示例中，您使用 AWS 账户 凭证来创建 IAM 用户。一开始，这些用户没有权限。您逐步为这些用户授予权限以执行特定的 Amazon S3 操作。为了测试这些权限，您使用每个用户的凭证登录到控制台。由于您不停地以 AWS 账户拥有者身份授予许可，并以 IAM 用户身份测试这些许可，因此需要登录和注销，每次都要使用不同的凭证。您可以使用一个浏览器执行该测试，但如果使用两个不同的浏览器，则可以加快该过程的速度。使用您的 AWS 账户 凭证通过一个浏览器连接到 AWS Management Console，并使用 IAM 用户凭证通过另一个浏览器进行连接。

要使用 AWS 账户 凭证登录 AWS Management Console，请转到 https://console.aws.amazon.com/。IAM 用户无法使用相同的链接登录。IAM 用户必须使用支持 IAM 的登录页面。作为账户所有者，您可以向您的用户提供此链接。

有关 IAM 的更多信息，请参阅 IAM 用户指南中的 AWS Management Console 登录页面。

向 IAM 用户提供登录链接的步骤

1. 登录 AWS Management Console，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在 Navigation (导航) 窗格中，请选择 IAM Dashboard (IAM 控制面板)。
3. 记下 IAM users sign in link: (IAM 用户登录链接：) 下的 URL。向 IAM 用户提供此链接，允许他们使用 IAM 用户名称和密码登录控制台。

步骤 1：创建存储桶

在该步骤中，您使用 AWS 账户 凭证登录到 Amazon S3 控制台，创建一个存储桶，在该存储桶中添加文件夹，然后在每个文件夹中上传一个或两个示例文档。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 创建存储桶。

   如需分步指导，请参阅 创建存储桶。
3. 将一个文档上传到存储桶。

   该练习假定您在该存储桶的根级别具有 `s3-dg.pdf` 文档。如果您上传其他文档，请使用其文件名替代 `s3-dg.pdf`。
4. 将三个名为 `Private`、`Finance` 和 `Development` 的文件夹添加到存储桶中。

有关创建文件夹的分步说明，请参阅 Amazon Simple Storage Service 用户指南中的 使用文件夹在 Amazon S3 控制台中整理对象>。

5. 将一个或两个文档上传到每个文件夹。

   对于该练习，假设您在每个文件夹中上传了几个文档，从而导致存储桶包含具有以下键的对象：

   - `Private/privDoc1.txt`
   - `Private/privDoc2.zip`
   - `Development/project1.xls`
   - `Development/project2.xls`
   - `Finance/Tax2011/document1.pdf`
   - `Finance/Tax2011/document2.pdf`
   - `s3-dg.pdf`

   如需分步指导，请参阅 上传对象。

步骤 2：创建 IAM 用户和组

现在使用 IAM 控制台将两个 IAM 用户（Alice 和 Bob）添加到您的 AWS 账户。有关分步说明，请参阅《IAM 用户指南》中的在您的 AWS 账户中创建 IAM 用户。

还要创建一个名为 `Consultants` 的管理组。然后，将这两个用户都添加到该组中。有关分步说明，请参阅创建 IAM 用户组。

> **Warning**
>
> 添加用户和组时，请不要为这些用户附加任何授予权限的策略。起初，这些用户没有任何权限。在以下部分中，您逐步授予权限。首先，您必须确保已为这些 IAM 用户分配密码。您使用这些用户凭证测试 Amazon S3 操作，并验证这些权限是否可按预期工作。

有关创建新的 IAM 用户的分步说明，请参阅《IAM 用户指南》中的在您的 AWS 账户中创建 IAM 用户。在为本演练创建用户时，请选择 AWS Management Console访问并清除编程访问。

有关创建管理组的分步说明，请参阅 IAM 用户指南中的创建第一个 IAM 管理员用户和组。

步骤 3：确认 IAM 用户没有任何权限

如果您使用两个浏览器，现在可以使用第二个浏览器通过 IAM 用户凭证之一登录控制台。

1. 通过使用 IAM 用户登录链接（请参阅向 IAM 用户提供登录链接的步骤），使用任一 IAM 用户凭证登录到 AWS Management Console。
2. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

   验证是否有告知您访问已被拒绝的控制台消息。

现在，您可以开始为用户授予增量权限。首先，您附加一个组策略，以授予两个用户必须具有的权限。

步骤 4：授予组级权限

您希望用户能够执行以下操作：

- 列出父账户拥有的所有存储桶。要执行此操作，Bob 和 Alice 必须拥有执行 `s3:ListAllMyBuckets` 操作的权限。
- 列出 `companybucket` 存储桶中的根级项目、文件夹和对象。要执行此操作，Bob 和 Alice 必须拥有在 `companybucket` 存储桶上执行 `s3:ListBucket` 操作的权限。

首先，您创建一个授予这些权限的策略，然后将其附加到 `Consultants` 组。

步骤 4.1：授予列出所有存储桶的权限

在该步骤中，您创建一个托管策略，它为用户授予最低权限，以使他们能够列出父账户拥有的所有存储桶。然后，您将该策略附加到 `Consultants` 组。在将该托管式策略附加到一个用户或组时，您为该用户或组授予权限以获取父 AWS 账户拥有的存储桶列表。

1. 登录 AWS Management Console，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

   > **Note**
   >
   > 由于您要授予用户权限，请使用您的 AWS 账户凭证登录，而不是以 IAM 用户身份登录。

2. 创建托管策略。

   a. 在左侧导航窗格中，请选择策略，然后选择创建策略。

b. 请选择 JSON 选项卡。

c. 复制以下访问策略，并将其粘贴到策略文本字段中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": ["s3:ListAllMyBuckets"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    }
  ]
}
```

策略是一个 JSON 文档。在文档中，`Statement` 是一个对象数组，其中每个对象均使用名称/值对的集合来描述权限。之前的策略描述了一个特定的权限。`Action` 指定访问权限的类型。在策略中，`s3:ListAllMyBuckets` 是预定义的 Amazon S3 操作。此操作涵盖了 Amazon S3 GET Service 操作，它可以返回已验证发件人拥有的所有存储桶的列表。`Effect` 元素值确定是允许还是拒绝特定的权限。

d. 选择 Review Policy（查看策略）。在下一页上，在名称字段中输入 `AllowGroupToSeeBucketListInTheConsole`，然后选择创建策略。

> **Note**
>
> 摘要条目显示一条消息，以指出该策略不授予任何权限。对于本演练，您可以安全地忽略此消息。

3. 将您创建的 `AllowGroupToSeeBucketListInTheConsole` 托管策略附加到 `Consultants` 组。

   有关附加托管式策略的分步说明，请参阅《IAM 用户指南》中的添加和删除 IAM 身份权限。

   您可以将策略文档附加到 IAM 控制台中的 IAM 用户和组。由于您希望两个用户都能够列出存储桶，因此，您将该策略附加到组。

4. 测试权限。

   a. 使用 IAM 用户登录链接（请参阅 向 IAM 用户提供登录链接的步骤），通过任何 IAM 用户凭证登录控制台。

b. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

控制台现在应该会列出所有的存储桶，但不会列出任何存储桶中的对象。

步骤 4.2：允许用户列出存储桶的根级内容

接下来，您允许 Consultants 组中的所有用户列出根级 companybucket 存储桶项目。当用户在 Amazon S3 控制台上选择 company 存储桶时，用户可以看到该存储桶中的根级项目。

> **Note**
>
> 该示例使用 companybucket 以进行说明。您必须使用创建的存储桶的名称。

要了解在您选择存储桶名称时控制台向 Amazon S3 发送的请求、Amazon S3 返回的响应，以及控制台如何解释该响应，请更详细地介绍该过程。

当您选择存储桶名称时，控制台会向 Amazon S3 发送 GET Bucket (List Objects)请求。该请求包括以下参数：

- 将空字符串作为值的 prefix 参数。
- 将 delimiter 作为值的 / 参数。

以下是一个示例请求。

```
GET ?prefix=&delimiter=/ HTTP/1.1
Host: companybucket.s3.amazonaws.com
Date: Wed, 01 Aug  2012 12:00:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Amazon S3 将返回包含以下 <ListBucketResult/> 元素的响应。

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix></Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>s3-dg.pdf</Key>
    ...
```

```
  </Contents>
  <CommonPrefixes>
    <Prefix>Development/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Finance/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Private/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

键 s3-dg.pdf 对象不包含斜杠 (/) 分隔符，并且 Amazon S3 在 <Contents> 元素中返回该键。但是，示例存储桶中的所有其他键都包含 / 分隔符。Amazon S3 将组合这些密钥，并在指定的 <CommonPrefixes> 分隔符首次出现时，为每个不同前缀值 Development/、Finance/ 和 Private/（它们是这些密钥开头的子字符串）返回 / 元素。

控制台将解释该结果并将根级项目显示为三个文件夹和一个对象键。

如果 Bob 或 Alice 打开 Development (开发) 文件夹，控制台将向 Amazon S3 发送 GET Bucket (List Objects) 请求，并将 prefix 和 delimiter 参数设置为以下值：

- 具有 prefix 值的 Development/ 参数。
- 具有“delimiter”值的 / 参数。

作为响应，Amazon S3 将返回对象密钥，该密钥以指定的前缀开头。

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix>Development</Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>Project1.xls</Key>
    ...
  </Contents>
  <Contents>
    <Key>Project2.xls</Key>
    ...
  </Contents>
</ListBucketResult>
```

控制台显示这些对象键。

现在，我们返回以授予用户权限，以便列出根级存储桶项目。要列出存储桶内容，用户需要调用 s3:ListBucket 操作的权限，如下面的策略语句中所述。为了确保他们仅看到根级内容，您添加一个条件以要求用户必须在请求中指定空 prefix，即不允许他们双击任何根级文件夹。最后，您添加一个条件以要求用户请求包含值为“delimiter”的 / 参数，从而要求进行文件夹样式的访问。

```
{
  "Sid": "AllowRootLevelListingOfCompanyBucket",
  "Action": ["s3:ListBucket"],
  "Effect": "Allow",
  "Resource": ["arn:aws:s3:::companybucket"],
  "Condition":{
         "StringEquals":{
             "s3:prefix":[""], "s3:delimiter":["/"]
                       }
          }
}
```

在 Amazon S3 控制台上选择一个存储桶时，控制台先发送 GET Bucket location 请求以查找部署了该存储桶的 AWS 区域。然后，控制台使用该存储桶的区域特定端点以发送 GET Bucket (List Objects)请求。因此，如果用户要使用控制台，您必须授予 s3:GetBucketLocation 操作权限，如以下策略语句中所示。

```
{
   "Sid": "RequiredByS3Console",
   "Action": ["s3:GetBucketLocation"],
   "Effect": "Allow",
   "Resource": ["arn:aws:s3:::*"]
}
```

允许用户列出根级存储桶内容的步骤

1. 登录 AWS Management Console，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

   使用 AWS 账户凭证，而不是 IAM 用户的凭证登录控制台。

2. 将附加到 Consultants 组的现有 AllowGroupToSeeBucketListInTheConsole 托管策略替换为以下策略，这还会允许执行 s3:ListBucket 操作。记得将策略 Resource 中的 *companybucket* 替换为您的存储桶的名称。

有关分步说明，请参阅《IAM 用户指南》中的编辑 IAM 策略。在按照分步说明进行操作时，请务必执行将更改应用于策略附加到的所有主体实体的步骤。

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
       "Action": [ "s3:ListAllMyBuckets", "s3:GetBucketLocation" ],
       "Effect": "Allow",
       "Resource": [ "arn:aws:s3:::*"  ]
     },
     {
       "Sid": "AllowRootLevelListingOfCompanyBucket",
       "Action": ["s3:ListBucket"],
       "Effect": "Allow",
       "Resource": ["arn:aws:s3:::companybucket"],
       "Condition":{
             "StringEquals":{
                    "s3:prefix":[""], "s3:delimiter":["/"]
                           }
                  }
     }
  ]
}
```

3. 测试更新的权限。

   a. 使用 IAM 用户登录链接（请参阅向 IAM 用户提供登录链接的步骤）登录 AWS Management Console。

      通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

   b. 请选择您创建的存储桶，控制台将显示根级存储桶项目。如果您选择存储桶中的任何文件夹，则无法看到文件夹内容，因为您尚未授予这些权限。

当用户使用 Amazon S3 控制台时，此测试会成功。在控制台上选择一个存储桶时，控制台实施发送一个请求，其中包含将空字符串作为值的 `prefix` 参数以及将“`delimiter`”作为值的 `/` 参数。

步骤 4.3：组策略总结

您所添加的组策略的实际结果是授予 IAM 用户 Alice 和 Bob 以下最低许可：

- 列出父账户拥有的所有存储桶。
- 查看 companybucket 存储桶中的根级项目。

不过，用户仍然只能执行很少的操作。接下来，您授予用户特定的权限，如下所示：

- 允许 Alice 在 Development 文件夹中获取和放置对象。
- 允许 Bob 在 Finance 文件夹中获取和放置对象。

对于用户特定的权限，您需要将策略附加到特定的用户，而不是附加到组。在以下部分中，您为 Alice 授予权限以在 Development 文件夹中工作。您可以重复这些步骤，以便为 Bob 授予类似权限以在 Finance 文件夹中工作。

步骤 5：授予 IAM 用户 Alice 特定的权限

现在，您为 Alice 授予额外的权限，以便她可以查看 Development 文件夹内容以及在该文件夹中获取和放置对象。

步骤 5.1：授予 IAM 用户 Alice 列出 development 文件夹内容的权限

要让 Alice 列出 Development 文件夹内容，您必须将一个策略应用于用户 Alice 以授予对 s3:ListBucket 存储桶执行 companybucket 操作的权限，但前提是请求包含 Development/ 前缀。您希望仅将该策略应用于 Alice 用户，因此，您使用一个内联策略。有关内联策略的更多信息，请参阅《IAM 用户指南》中的 托管式策略与内联策略。

1. 登录 AWS Management Console，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

   使用 AWS 账户凭证，而不是 IAM 用户的凭证登录控制台。
2. 创建一个内联策略，以便为 Alice 用户授予权限以列出 Development 文件夹内容。

   a. 在左侧导航窗格中，请选择用户。
   b. 请选择用户名 Alice。
   c. 在用户详细信息页面上，请选择权限选项卡，然后选择添加内联策略。
   d. 请选择 JSON 选项卡。

e. 复制以下策略，并将其粘贴到策略文本字段中。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
      "Sid": "AllowListBucketIfSpecificPrefixIsIncludedInRequest",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{  "StringLike":{"s3:prefix":["Development/*"] }
       }
    }
  ]
}
```

f. 选择 Review Policy（查看策略）。在下一页上，在名称字段中输入一个名称，然后选择创建策略。

3. 测试对 Alice 的权限的更改：

a. 使用 IAM 用户登录链接（请参阅向 IAM 用户提供登录链接的步骤）登录 AWS Management Console。

b. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

c. 在 Amazon S3 控制台中，验证 Alice 能否查看存储桶中 Development/ 文件夹中的对象列表。

当用户单击 /Development 文件夹以查看其中的对象列表时，Amazon S3 控制台会向 Amazon S3 发送 ListObjects 请求并使用前缀 /Development。由于用户已被授予许可，可以查看使用前缀 Development 和分隔符 / 的对象列表，因此 Amazon S3 将返回使用密钥前缀 Development/ 的对象列表，并且控制台将显示该列表。

步骤 5.2：授予 IAM 用户 Alice 在 development 文件夹中获取和放置对象的权限

要让 Alice 在 Development 文件夹中获取和放置对象，她需要具有调用 s3:GetObject 和 s3:PutObject 操作的权限。以下策略语句授予这些权限，但前提是请求包含值为 prefix 的 Development/ 参数。

```
{
    "Sid":"AllowUserToReadWriteObjectData",
```

```
    "Action":["s3:GetObject", "s3:PutObject"],
    "Effect":"Allow",
    "Resource":["arn:aws:s3:::companybucket/Development/*"]
  }
```

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

   使用 AWS 账户凭证，而不是 IAM 用户的凭证登录控制台。

2. 编辑您在上一步中创建的内联策略。

   a. 在左侧导航窗格中，请选择用户。

   b. 请选择 Alice 用户名。

   c. 在用户详细信息页面上，请选择权限选项卡，然后展开内联策略部分。

   d. 在上一步中创建的策略名称旁边，请选择编辑策略。

   e. 复制以下策略，并将其粘贴到策略文本字段中，从而替换现有的策略。

```
{
     "Version": "2012-10-17",
     "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      }
   ]
}
```

3. 测试更新的策略：

a. 使用 IAM 用户登录链接（请参阅向 IAM 用户提供登录链接的步骤）登录 AWS Management Console。
b. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
c. 在 Amazon S3 控制台中，验证 Alice 现在能否在 Development 文件夹中添加和下载对象。

步骤 5.3：显式拒绝 IAM 用户 Alice 对存储桶中其他任何文件夹的权限

用户 Alice 现在可以列出 companybucket 存储桶中的根级内容。她还可以在 Development 文件夹中获取和放置对象。如果您确实想要严格控制访问权限，可以显式拒绝 Alice 对存储桶中其他任何文件夹的访问。若存在任何其他策略（存储桶策略或 ACL），而且它们授予了 Alice 访问存储桶中任何其他文件夹的权限，此显式拒绝将覆盖这些权限。

您可以向用户 Alice 策略添加以下声明：要求 Alice 发送到 Amazon S3 的所有请求均包含 prefix 参数，该参数的值可为 Development/* 或空字符串。

```
{
   "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
   "Action": ["s3:ListBucket"],
   "Effect": "Deny",
   "Resource": ["arn:aws:s3:::companybucket"],
   "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                  "Null"         : {"s3:prefix":false }
    }
}
```

在 Condition 块中具有两个条件表达式。这些条件表达式的结果将使用逻辑 AND 合并在一起。如果两种条件均为真，组合条件的结果才为真。由于该策略中的 Effect 为 Deny，因此，在 Condition 计算结果为 true 时，用户无法执行指定的 Action。

- Null 条件表达式确保来自 Alice 的请求包含 prefix 参数。

  prefix 参数要求类似文件夹的访问权限。如果您发送的请求没有包含 prefix 参数，Amazon S3 将返回所有的对象密钥。

  如果请求包含具有空值的 prefix 参数，则表达式的计算结果为 true，因此，整个 Condition 的计算结果为 true。您必须允许将空字符串作为 prefix 参数的值。在前面讨论的内容中，回顾了采用与前面讨论中控制台相同的方式，通过允许空字符串使 Alice 能够检索根级存储桶项目。有关更多信息，请参阅 步骤 4.2：允许用户列出存储桶的根级内容。

- StringNotLike 条件表达式确保在指定了 prefix 参数值并且不是 Development/* 时，请求失败。

执行上一节中的步骤，并再次更新为 Alice 用户创建的内联策略。

复制以下策略，并将其粘贴到策略文本字段中，从而替换现有的策略。

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      },
      {
         "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
         "Action": ["s3:ListBucket"],
         "Effect": "Deny",
         "Resource": ["arn:aws:s3:::companybucket"],
         "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                        "Null"         : {"s3:prefix":false }
          }
      }
   ]
}
```

步骤 6：授予 IAM 用户 Bob 特定的权限

现在，您希望为 Bob 授予 Finance 文件夹的权限。执行您之前为 Alice 授予权限时使用的步骤，但将 Development 文件夹替换为 Finance 文件夹。如需分步指导，请参阅 步骤 5：授予 IAM 用户 Alice 特定的权限。

步骤 7：确保 Private 文件夹的安全

在此示例中，您有两个用户。您已授予了所需最低组级权限，仅当确实需要单个用户级别的权限时才授予用户级权限。此方法有助于最大程度地降低管理权限的工作量。随着用户数的增加，管理权限工作将变得极为繁重。例如，您不希望该示例中的任何用户访问 Private 文件夹内容。您如何确保不会意外地为用户授予 Private 文件夹的权限？ 您可以添加一个策略，显式拒绝对该文件夹的访问。显式拒绝将覆盖任何其他权限。

要确保 Private 文件夹保密，您可以将以下两个拒绝语句添加到组策略中：

- 添加以下声明以显式拒绝对 Private 文件夹 (companybucket/Private/*) 中的资源执行任何操作。

```
{
  "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
  "Action": ["s3:*"],
  "Effect": "Deny",
  "Resource":["arn:aws:s3:::companybucket/Private/*"]
}
```

- 若请求指定了 Private/ 前缀，您也可以拒绝列出对象操作所需的权限。在控制台上，如果 Bob 或 Alice 打开 Private 文件夹，该策略导致 Amazon S3 返回错误响应。

```
{
  "Sid": "DenyListBucketOnPrivateFolder",
  "Action": ["s3:ListBucket"],
  "Effect": "Deny",
  "Resource": ["arn:aws:s3:::*"],
  "Condition":{
      "StringLike":{"s3:prefix":["Private/"]}
   }
}
```

将 Consultants 组策略替换为包含上述拒绝语句的更新策略。在应用更新的策略后，组中的任何用户都无法访问存储桶中的 Private 文件夹。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

   使用 AWS 账户凭证，而不是 IAM 用户的凭证登录控制台。

2. 将附加到 Consultants 组的现有 AllowGroupToSeeBucketListInTheConsole 托管策略替换为以下策略。记得将策略中的 *companybucket* 替换为您的存储桶的名称。

   有关说明，请参阅《IAM 用户指南》中的编辑客户托管策略。按照说明操作时，请确保按照指示操作，以便将您的更改应用到策略附加到的所有主体实体。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    },
    {
      "Sid": "AllowRootLevelListingOfCompanyBucket",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{
          "StringEquals":{"s3:prefix":[""]}
       }
    },
    {
      "Sid": "RequireFolderStyleList",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringNotEquals":{"s3:delimiter":"/"}
       }
     },
    {
      "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
      "Action": ["s3:*"],
      "Effect": "Deny",
```

```
      "Resource":["arn:aws:s3:::companybucket/Private/*"]
    },
    {
      "Sid": "DenyListBucketOnPrivateFolder",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringLike":{"s3:prefix":["Private/"]}
       }
    }
  ]
}
```

步骤 8：清除

要进行清除，请打开 IAM 控制台并移除用户 Alice 和 Bob。有关分步说明，请参阅《IAM 用户指南》中的删除 IAM 用户。

为了确保不再对您收取存储费用，您还应删除为该练习创建的对象和存储桶。

相关资源

- 《IAM 用户指南》中的 管理 IAM policy

演练：使用策略管理针对 Amazon S3 资源的访问权限

本主题提供了以下介绍性示例演练，演示如何授予针对 Amazon S3 资源的访问权限。这些示例使用 AWS Management Console 来创建资源（存储桶、对象、用户）并授予它们相应的权限。然后这些示例将向您演示如何使用命令行工具来验证这些权限而不必编写任何代码。我们使用 AWS Command Line Interface（AWS CLI）和 AWS Tools for Windows PowerShell 提供命令。

- 示例 1：存储桶拥有者向其用户授予存储桶权限

  您在您的账户中创建的 IAM 用户默认情况下没有权限。在本练习中，您要授予用户权限来执行存储桶和对象操作。

- 示例 2：存储桶拥有者授予跨账户存储桶权限

  在本练习中，存储桶拥有者账户 A 对另一个 AWS 账户（账户 B）授予跨账户权限，然后账户 B 将这些权限委派给其账户中的用户。

- 在对象与存储桶拥有者不同的情况下管理对象权限

  在本例中的示例方案是一个存储桶拥有者向其他人授予对象权限，但并不是该存储桶中所有对象都归该存储桶拥有者所有。存储桶拥有者需要什么权限，以及如何能委派这些权限？

  创建存储桶的 AWS 账户称为 存储桶拥有者。该拥有者可以向其他 AWS 账户授予上传对象的权限，而创建对象的 AWS 账户拥有该对象。存储桶拥有者对其他 AWS 账户创建的对象不拥有权限。如果该存储桶拥有者编写了一个存储桶策略来授予针对对象的访问权限，则该策略不适用于其它账户拥有的对象。

  在这种情况下，对象拥有者必须首先使用对象 ACL 向存储桶拥有者授予权限。然后存储桶拥有者才能够如以下示例所示，将这些对象权限委派给其他人、其自己账户中的用户或另一个 AWS 账户。

  - 示例 3：存储桶拥有者授予不属于自己的对象的权限

    在本练习中，存储桶拥有者首先从对象拥有者获取权限。然后存储桶拥有者将这些权限委派给自己的账户中的用户。

  - 示例 4：存储桶拥有者针对自己未拥有的对象授予跨账户权限

    在从对象拥有者获得权限后，存储桶拥有者无法将权限委派给其它 AWS 账户，因为不支持跨账户委派（请参阅授予权限）。但是，存储桶拥有者可以创建具有执行特定操作（如 get object）的权限的 IAM 角色，并允许其他 AWS 账户担任该角色。这样，任何担任该角色的人都能够访问对象。此示例显示存储桶拥有者如何使用 IAM 角色来启用跨账户委派。

在尝试示例演练之前

这些示例使用 AWS Management Console 来创建资源和授予权限。为了测试权限，这些示例使用命令行工具 AWS CLI 和 AWS Tools for Windows PowerShell，因此您无需编写任何代码。要测试权限，您必须设置其中的一个工具。有关更多信息，请参阅 设置用于演练的工具。

此外，在创建资源时，这些示例并未使用 AWS 账户的根用户凭证。而是在这些账户中创建一个管理员用户来执行这些任务。

关于使用管理员用户来创建资源和授予权限

AWS Identity and Access Management（IAM）建议不要使用 AWS 账户 的根用户凭证发起请求。而是应创建 IAM 用户或角色，向他们授予完全访问权限，然后使用其凭证来发出请求。我们将其称为管理员用户或角色。有关更多信息，请转至《AWS 一般参考》中的 AWS 账户根用户 凭证和 IAM 身份以及《IAM 用户指南》中的 IAM 最佳实践。

本部分中的所有示例演练都使用管理员用户凭证。如果您还未创建您的 AWS 账户的管理员用户，此处的主题会向您演示这一过程。

要使用用户凭证登录 AWS Management Console，您必须使用 IAM 用户登录 URL。IAM 控制台会为您的 AWS 账户提供此 URL。这些主题向您演示如何获取该 URL。

设置用于演练的工具

介绍性示例（请参阅 演练：使用策略管理针对 Amazon S3 资源的访问权限）使用 AWS Management Console 来创建资源和授予权限。为了测试权限，这些示例使用命令行工具 AWS Command Line Interface（AWS CLI）和 AWS Tools for Windows PowerShell，因此您无需编写任何代码。要测试权限，您必须设置其中的一个工具。

设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   安装或更新到最新版本的 AWS Command Line Interface

   开始使用 AWS Command Line Interface
2. 设置默认的配置文件。

   您将用户凭证存储在 AWS CLI config 文件中。使用 AWS 账户凭证在配置文件中创建默认配置。有关查找和编辑 AWS CLI 配置文件的说明，请参阅 Configuration and credential file settings。

```
[default]
aws_access_key_id = access key ID
aws_secret_access_key = secret access key
region = us-west-2
```

3. 在命令提示符处输入以下命令来验证设置。这两个命令没有显式提供凭证，因此将使用默认配置的凭证。

   - 尝试 help 命令。

     ```
     aws help
     ```

   - 要获取所配置账户的存储桶列表，请使用 aws s3 ls 命令。

     ```
     aws s3 ls
     ```

当您进行演练时，您将创建用户，并且通过创建配置将用户凭证保存在配置文件中，如以下示例所示。这些配置文件的名称为 AccountAadmin 和 AccountBadmin。

```
[profile AccountAadmin]
aws_access_key_id = User AccountAadmin access key ID
aws_secret_access_key = User AccountAadmin secret access key
region = us-west-2

[profile AccountBadmin]
aws_access_key_id = Account B access key ID
aws_secret_access_key = Account B secret access key
region = us-east-1
```

要使用这些用户凭证来运行命令，请添加 --profile 参数来指定配置文件名称。以下 AWS CLI 命令检索 *examplebucket* 中对象的列表，并指定 AccountBadmin 配置文件。

```
aws s3 ls s3://examplebucket --profile AccountBadmin
```

或者，您也可以从命令提示符处更改 AWS_DEFAULT_PROFILE 环境变量，将一组用户凭证配置为默认的配置文件。完成此操作后，当执行不带 --profile 参数的 AWS CLI 命令时，AWS CLI 会将您在环境变量中设置的配置文件用作默认配置文件。

```
$ export AWS_DEFAULT_PROFILE=AccountAadmin
```

要设置 AWS Tools for Windows PowerShell，请按以下步骤操作：

1. 下载并配置 AWS Tools for Windows PowerShell。有关说明，请参阅《AWS Tools for Windows PowerShell User Guide》中的 Installing the AWS Tools for Windows PowerShell。

   > **Note**
   >
   > 要加载 AWS Tools for Windows PowerShell 模块，必须启用 PowerShell 脚本执行。有关更多信息，请参阅《AWS Tools for Windows PowerShell User Guide》中的 Enable Script Execution。

2. 对于这些演练，您需使用 `Set-AWSCredentials` 命令指定每个会话的 AWS 凭证。该命令会将凭证保存到持久存储 (`-StoreAs` 参数)。

   ```
   Set-AWSCredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -
   storeas string
   ```

3. 验证设置。

   - 要检索可用于 Amazon S3 操作的命令的列表，请运行 `Get-Command` 命令。

     ```
     Get-Command -module awspowershell -noun s3* -StoredCredentials string
     ```

   - 要检索存储桶中的对象的列表，请运行 `Get-S3Object` 命令。

     ```
     Get-S3Object -BucketName bucketname -StoredCredentials string
     ```

有关命令列表，请参阅 AWS Tools for PowerShell Cmdlet Reference。

现在您已经准备就绪，可以尝试进行演练了。访问每个部分开头提供的链接。

示例 1：存储桶拥有者向其用户授予存储桶权限

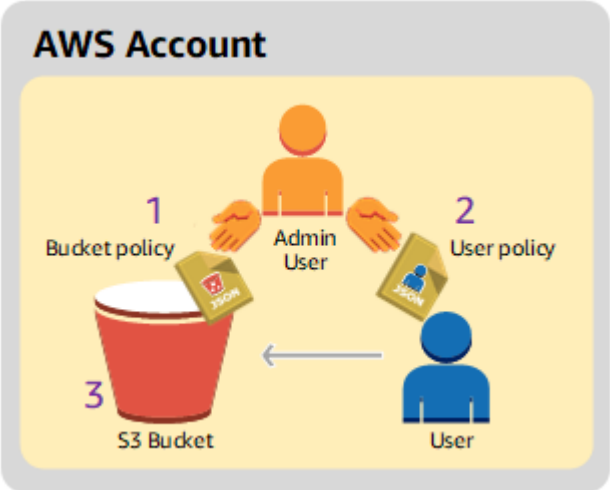
> **Important**
>
> 向 IAM 角色授予权限比向各个用户授予权限更好。有关如何向 IAM 角色授予权限的更多信息，请参阅了解跨账户权限和使用 IAM 角色。

主题

- 准备演练
- 步骤 1：在账户 A 中创建资源并授予权限
- 步骤 2：测试权限

在本演练中，一个 AWS 账户拥有存储桶，该账户中包含一个 IAM 用户。默认情况下，该用户没有权限。要让该用户执行任何任务，父账户必须向该用户授予权限。存储桶拥有者和父账父相同。因此，要向用户授予存储桶权限，AWS 账户可以使用存储桶策略和/或用户策略。存储桶拥有者将使用存储桶策略授予一些权限，使用用户策略授予其他权限。

以下是演练的步骤概括：



1. 账户管理员创建存储桶策略，向用户授予一组权限。
2. 账户管理员将用户策略附加到用户，授予其他权限。
3. 然后，用户尝试使用通过存储桶策略和用户策略授予的权限。

对于此示例，您需要一个 AWS 账户。您将创建一个管理员用户（请参阅 关于使用管理员用户来创建资源和授予权限），而不是使用账户的根用户凭证。我们按下表所示引用 AWS 账户和管理员用户：

账户 ID	账户名称	账户中的管理员用户
1111-1111-1111	账户 A	AccountAadmin

> **Note**
>
> 本示例中的管理员用户是 AccountAadmin，它代表账户 A，而不是 AccountAdmin。

创建用户和授予权限的所有任务都在 AWS Management Console 中完成。为验证权限，演练中使用命令行工具、AWS Command Line Interface (AWS CLI) 和 AWS Tools for Windows PowerShell，因此您无需编写任何代码。

准备演练

1. 确保您有一个 AWS 账户并且它的一个用户拥有管理员权限。

   a. 根据需要注册 AWS 账户。我们将此账户称为账户 A。

      i. 转到 https://aws.amazon.com/s3，然后单击创建 AWS 账户。

      ii. 按照屏幕上的说明进行操作。

      账户激活且可用时，AWS 会通过电子邮件向您发送通知。

   b. 在账户 A 中，创建一个管理员用户 **AccountAadmin**。使用账户 A 凭证登录 IAM 控制台，然后执行以下操作：

      i. 创建用户 **AccountAadmin** 并记下用户安全凭证。

      有关说明，请参阅《IAM 用户指南》中的在您的 AWS 账户中创建 IAM 用户。

      ii. 通过附加一个授予完全访问权限的用户策略来向 AccountAadmin 授予管理员权限。

      有关说明，请参阅《IAM 用户指南》中的管理 IAM 策略。

      iii. 记下 AccountAadmin 的 IAM 用户登录 URL。您在登录 AWS Management Console时需要使用此 URL。有关如何查找登录 URL 的更多信息，请参阅《IAM 用户指南》中的 Sign in to the AWS Management Console as an IAM user。记下每个账户的 URL。

2. 设置 AWS CLI 或 AWS Tools for Windows PowerShell。请务必保存管理员用户凭证，如下所示：

   - 如果使用 AWS CLI，请在配置文件中创建配置文件 `AccountAadmin`。
   - 如果使用 AWS Tools for Windows PowerShell，请确保将用于会话的凭证存储为 `AccountAadmin`。

   有关说明，请参阅 设置用于演练的工具。

步骤 1：在账户 A 中创建资源并授予权限

使用账户 A 中用户 `AccountAadmin` 的凭证和专用的 IAM 用户登录 URL 登录 AWS Management Console，然后执行以下操作：

1. 创建资源（一个存储桶和一个 IAM 用户）

   a. 在 Amazon S3 控制台中创建一个存储桶。记下创建存储桶的 AWS 区域。有关说明，请参阅创建存储桶。

   b. 在 IAM 控制台中，执行以下操作：

      i. 创建一个名为 Dave 的用户。

         有关分步说明，请参阅《IAM 用户指南》中的创建 IAM 用户（控制台）。

      ii. 记下 `UserDave` 凭证。

      iii. 记下用户 Dave 的 Amazon 资源名称（ARN）。在 IAM 控制台中，选择该用户，摘要选项卡会提供用户 ARN。

2. 授予权限

   因为存储桶拥有者和用户所属的父账户相同，所以 AWS 账户可以使用存储桶策略和/或用户策略向用户授予权限。在此示例中，您同时使用这两种方法。如果对象也由同一个账户拥有，则存储桶拥有者可以在存储桶策略（或 IAM 策略）中授予对象权限。

   a. 在 Amazon S3 控制台中，将以下存储桶策略附加到 *`awsexamplebucket1`*。

      该策略包含两个语句。

      - 第一个语句向 Dave 授予存储桶操作权限 `s3:GetBucketLocation` 和 `s3:ListBucket`。

- 第二个语句授予 s3:GetObject 权限。因为账户 A 还拥有对象，所以账户管理员能够授予 s3:GetObject 权限。

在 Principal 语句中，Dave 通过其用户 ARN 进行标识。有关策略元素的更多信息，请参阅 Amazon S3 中的策略和权限。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "statement1",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
         },
         "Action": [
            "s3:GetBucketLocation",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1"
         ]
      },
      {
         "Sid": "statement2",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
         },
         "Action": [
             "s3:GetObject"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*"
         ]
      }
   ]
}
```

b. 使用以下策略为用户 Dave 创建一个内联策略。该策略向 Dave 授予 s3:PutObject 权限。您需要通过提供存储桶名称来更新策略。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "PermissionForObjectOperations",
         "Effect": "Allow",
         "Action": [
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*"
         ]
      }
   ]
}
```

有关说明，请参阅《IAM 用户指南》中的管理 IAM 策略。请注意，您需要使用账户 A 凭证登录控制台。

步骤 2：测试权限

使用 Dave 的凭证验证权限是否发挥作用。您可以使用以下两个过程之一。

使用 AWS CLI 测试权限

1. 通过添加以下 UserDaveAccountA 配置文件更新 AWS CLI 配置文件。有关更多信息，请参阅设置用于演练的工具。

   ```
   [profile UserDaveAccountA]
   aws_access_key_id = access-key
   aws_secret_access_key = secret-access-key
   region = us-east-1
   ```

2. 验证 Dave 是否可以执行在用户策略中授权的操作。使用以下 AWS CLI put-object 命令上传示例对象。

   该命令中的 --body 参数指示要上传的源文件。例如，如果文件处于 Windows 计算机上 C: 驱动器的根目录中，则指定 c:\HappyFace.jpg。--key 参数提供对象的键名。

```
aws s3api put-object --bucket awsexamplebucket1 --key HappyFace.jpg --
body HappyFace.jpg --profile UserDaveAccountA
```

运行以下 AWS CLI 命令以获取对象。

```
aws s3api get-object --bucket awsexamplebucket1 --key HappyFace.jpg OutputFile.jpg
 --profile UserDaveAccountA
```

使用 AWS Tools for Windows PowerShell 测试权限

1. 将 Dave 的凭证存储为 AccountADave。您随后使用这些凭证对对象执行 PUT 和 GET 操作。

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountADave
```

2. 通过用户 Dave 的已存储凭证，使用 AWS Tools for Windows PowerShell Write-S3Object 命令上传示例对象。

```
Write-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file HappyFace.jpg
 -StoredCredentials AccountADave
```

下载之前上传的对象。

```
Read-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file Output.jpg -
StoredCredentials AccountADave
```

示例 2：存储桶拥有者授予跨账户存储桶权限

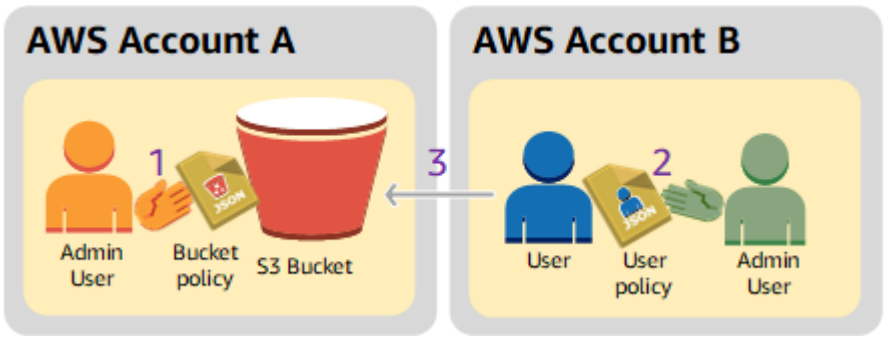
> **Important**
>
> 向 IAM 角色授予权限是比向各个用户授予权限更好的做法。若要了解如何执行此操作，请参阅了解跨账户权限和使用 IAM 角色。

主题

一个 AWS 账户（例如账户 A）可以将访问自身资源（如存储桶和对象）的权限授予另一个 AWS 账户（账户 B）。账户 B 随后可以将这些权限委托给其账户中的用户。在此示例情景中，存储桶拥有者向另一个账户授予执行特定存储桶操作的跨账户权限。

> **Note**
>
> 账户 A 还可以使用存储桶策略直接向账户 B 中的用户授予权限。但是，该用户仍需要来自用户所属的父账户（账户 B）的权限，即使账户 B 没有来自账户 A 的权限也是如此。该用户只要同时拥有来自资源拥有者和父账户的权限，便能够访问资源。

下面概括介绍演练步骤：



1. 账户 A 管理员用户附加一个存储桶策略，该策略向账户 B 授予执行特定存储桶操作的跨账户权限。

请注意，账户 B 中的管理员用户将自动继承这些权限。

2. 账户 B 管理员用户将用户策略附加到用户，委托从账户 A 收到的权限。
3. 账户 B 中的用户随后通过访问账户 A 拥有的存储桶中的对象来验证权限。

对于此示例，您需要两个账户。下表显示我们如何引用这些账户和其中的管理员用户。根据 IAM 准则（请参阅关于使用管理员用户来创建资源和授予权限），我们在本次演练中并不使用根用户凭证。而是在每个账户中创建一个管理员用户，并在创建资源和向他们授予权限时使用这些凭证。

AWS 账户 ID	账户名称	账户中的管理员用户
1111-1111-1111	账户 A	AccountAadmin
2222-2222-2222	账户 B	AccountBadmin

创建用户和授予权限的所有任务都在 AWS Management Console 中完成。为验证权限，演练中使用命令行工具、AWS Command Line Interface (CLI) 和 AWS Tools for Windows PowerShell，因此您无需编写任何代码。

准备演练

1. 确保您有两个 AWS 账户并且每个账户都有一个管理员用户，如前面部分的表中所示。

   a. 根据需要注册 AWS 账户。

   b. 使用账户 A 凭证登录 IAM 控制台以创建管理员用户：

      i. 创建用户 **`AccountAadmin`** 并记下安全凭证。有关说明，请参阅《IAM 用户指南》中的在您的 AWS 账户 中创建 IAM 用户。

      ii. 通过附加一个授予完全访问权限的用户策略来向 AccountAadmin 授予管理员权限。有关说明，请参阅 IAM 用户指南中的使用策略。

   c. 在 IAM 控制台中，记下控制面板上的 IAM 用户登录 URL。账户中的所有用户都必须使用此 URL 登录 AWS Management Console。

      有关更多信息，请参阅 IAM 用户指南中的用户如何登录您的账户。

   d. 使用账户 B 凭证重复上一个步骤，然后创建管理员用户 **`AccountBadmin`**。

2. 设置 AWS Command Line Interface（AWS CLI）或 AWS Tools for Windows PowerShell。请务必保存管理员用户凭证，如下所示：

- 如果使用 AWS CLI，请在配置文件中创建两个配置文件：AccountAadmin 和 AccountBadmin。
- 如果使用 AWS Tools for Windows PowerShell，请确保将用于会话的凭证存储为 AccountAadmin 和 AccountBadmin。

有关说明，请参阅 设置用于演练的工具。

3. 保存管理员用户凭证，也称为配置文件。您可以对输入的每个命令使用配置文件名称而不是指定凭证。有关更多信息，请参阅 设置用于演练的工具。

   a. 在 AWS CLI 凭证文件中为两个账户中的每个管理员用户 AccountAadmin 和 AccountBadmin 添加配置文件。

```
[AccountAadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1

[AccountBadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1
```

   b. 如果使用的是 AWS Tools for Windows PowerShell，请运行以下命令。

```
set-awscredentials -AccessKey AcctA-access-key-ID -SecretKey AcctA-secret-
access-key -storeas AccountAadmin
set-awscredentials -AccessKey AcctB-access-key-ID -SecretKey AcctB-secret-
access-key -storeas AccountBadmin
```

步骤 1：执行账户 A 任务

步骤 1.1：登录到 AWS Management Console

首先使用账户 A 的 IAM 用户登录 URL，以 AccountAadmin 用户身份登录 AWS Management Console。此用户将创建一个存储桶并向其附加一个策略。

步骤 1.2：创建存储桶

1. 在 Amazon S3 控制台中创建一个存储桶。此练习假设该存储桶在美国东部（弗吉尼亚州北部）AWS 区域中创建，命名为 *amzn-s3-demo-bucket*。

   有关说明，请参阅 创建存储桶。

2. 向存储桶上传示例对象。

   有关说明，请转到 步骤 2：将对象上传到存储桶。

步骤 1.3：附加一个存储桶策略，向账户 B 授予跨账户权限

该存储桶策略向账户 B 授予 s3:GetLifecycleConfiguration 和 s3:ListBucket 权限。假设您仍使用 AccountAadmin 用户凭证登录控制台。

1. 将以下存储桶策略附加到 *amzn-s3-demo-bucket*。该策略向账户 B 授予 s3:GetLifecycleConfiguration 和 s3:ListBucket 操作的权限。

   有关说明，请参阅 使用 Amazon S3 控制台添加存储桶策略。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

2. 验证账户 B（及其管理员用户）是否可以执行这些操作。

- 使用 AWS CLI 进行验证

```
aws s3 ls s3://amzn-s3-demo-bucket --profile AccountBadmin
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --
profile AccountBadmin
```

- 使用 AWS Tools for Windows PowerShell 进行验证

```
get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBadmin
get-s3bucketlifecycleconfiguration -BucketName amzn-s3-demo-bucket -
StoredCredentials AccountBadmin
```

步骤 2：执行账户 B 任务

现在账户 B 管理员创建用户 Dave，并委派从账户 A 收到的权限。

步骤 2.1：登录到 AWS Management Console

首先使用账户 B 的 IAM 用户登录 URL，以 AccountBadmin 用户身份登录 AWS Management Console。

步骤 2.2：在账户 B 中创建用户 Dave

在 IAM 控制台中，创建用户 **Dave**。

有关说明，请参阅 IAM 用户指南中的创建 IAM 用户（控制台）。

步骤 2.3：向用户 Dave 授予权限

使用以下策略为用户 Dave 创建一个内联策略。您需要通过提供存储桶名称来更新策略。

假设您使用 AccountBadmin 用户凭证登录控制台。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example",
         "Effect": "Allow",
         "Action": [
```

```
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

有关说明，请参阅《IAM 用户指南》中的管理 IAM 策略。

步骤 2.4：测试权限

现在，账户 B 中的 Dave 可以列出账户 A 拥有的 *amzn-s3-demo-bucket* 的内容。您可以使用以下过程之一验证权限。

使用 AWS CLI 测试权限

1. 将 UserDave 配置文件添加到 AWS CLI 配置文件。有关配置文件的更多信息，请参阅设置用于演练的工具。

   ```
   [profile UserDave]
   aws_access_key_id = access-key
   aws_secret_access_key = secret-access-key
   region = us-east-1
   ```

2. 在命令提示符处，输入以下 AWS CLI 命令，验证 Dave 现在是否可以从账户 A 拥有的 *amzn-s3-demo-bucket* 获取对象列表。请注意，该命令指定了 UserDave 配置文件。

   ```
   aws s3 ls s3://amzn-s3-demo-bucket --profile UserDave
   ```

   Dave 没有任何其它权限。因此，如果他尝试任何其它操作（例如，下面的 get-bucket-lifecycle 配置操作），则 Amazon S3 会返回权限被拒绝的消息。

   ```
   aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --profile
    UserDave
   ```

使用 AWS Tools for Windows PowerShell 测试权限

1. 将 Dave 的凭证存储为 AccountBDave。

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountBDave
```

2. 尝试列出存储桶的命令。

```
get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBDave
```

Dave 没有任何其它权限。因此，如果他尝试任何其它操作（例如，下面的 get-s3bucketlifecycleconfiguration 操作），则 Amazon S3 会返回权限被拒绝的消息。

```
get-s3bucketlifecycleconfiguration -BucketName amzn-s3-demo-bucket -
StoredCredentials AccountBDave
```

步骤 3：（可选）尝试显式拒绝

您可以使用访问控制列表（ACL）、存储桶策略或用户策略授予权限。但是，如果通过存储桶策略或用户策略设置了显式拒绝，则显式拒绝优先于任何其它权限。为进行测试，请更新存储桶策略，对账户 B 显式拒绝 s3:ListBucket 权限。该策略还授予了 s3:ListBucket 权限。但是，显式拒绝处于优先地位，因此账户 B 或账户 B 中的用户无法列出 *amzn-s3-demo-bucket* 中的对象。

1. 使用账户 A 中的用户 AccountAadmin 的凭证，将存储桶策略替换为以下内容。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      },
```

```
      {
         "Sid": "Deny permission",
         "Effect": "Deny",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

2. 现在，如果您尝试使用 AccountBadmin 凭证获取存储桶列表，则您的访问会被拒绝。

   - 使用 AWS CLI，运行以下命令：

     ```
     aws s3 ls s3://amzn-s3-demo-bucket --profile AccountBadmin
     ```

   - 使用 AWS Tools for Windows PowerShell，运行以下命令：

     ```
     get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBDave
     ```

步骤 4：清除

1. 完成测试之后，您可以执行以下操作来进行清理：

   - 使用账户 A 凭证登录 AWS Management Console（AWS Management Console），执行以下操作：

     - 在 Amazon S3 控制台中，删除附加到 *amzn-s3-demo-bucket* 的存储桶策略。在存储桶 Properties（属性）中，删除 Permissions（权限）部分中的策略。
     - 如果该存储桶是为此练习而创建的，请在 Amazon S3 控制台中删除对象，然后删除存储桶。
     - 在 IAM 控制台中，移除 AccountAadmin 用户。

2. 使用账户 B 凭证登录 IAM 控制台。删除用户 AccountBadmin。有关分步说明，请参阅《IAM 用户指南》中的删除 IAM 用户。

示例 3：存储桶拥有者授予不属于自己的对象的权限

> **Important**
>
> 向 IAM 角色授予权限是比向各个用户授予权限更好的做法。若要了解如何执行此操作，请参阅了解跨账户权限和使用 IAM 角色。

主题

此示例的情况是，存储桶拥有者要授予对象访问权限，但该存储桶拥有者并未拥有该存储桶中的所有对象。在此示例中，存储桶拥有者想要向其账户中的用户授予权限。

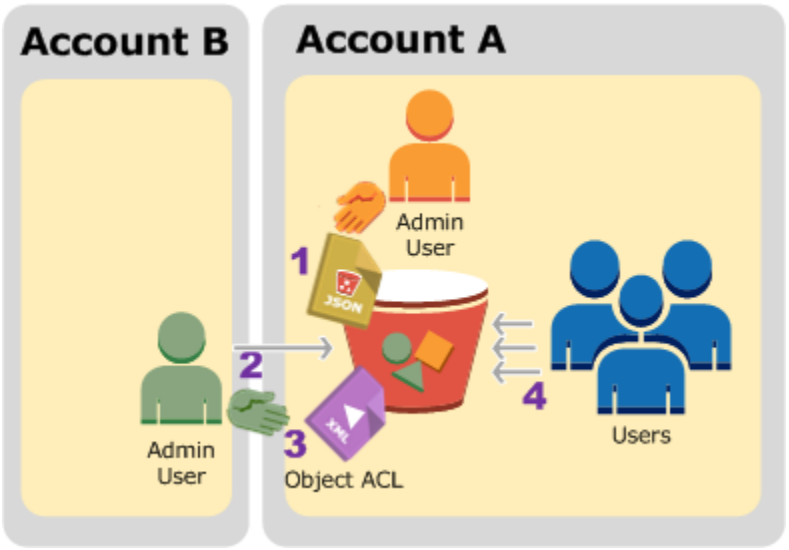
存储桶拥有者可让其他 AWS 账户上传对象。默认情况下，存储桶拥有者不拥有 AWS 账户 写入存储桶的对象。对象归将它们写入 S3 存储桶的账户所有。如果存储桶拥有者未拥有存储桶中的对象，对象拥有者必须首先使用对象访问控制列表（ACL）授予存储桶拥有者权限。然后，存储桶拥有者可以授予他们未拥有的对象的权限。有关更多信息，请参阅 Amazon S3 存储桶和对象所有权。

如果存储桶拥有者对存储桶的 S3 对象所有权应用了强制存储桶拥有者设置，则存储桶拥有者将拥有存储桶中的所有对象，包括另一个 AWS 账户 所编写的对象。此方法将解决存储桶拥有者未拥有对象的问题。然后，您可以将权限委派给自己账户中的用户或其他 AWS 账户。

> **Note**
>
> S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权和禁用或启用 ACL。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对这些对象的访问权限。
>
> Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有需要单独控制每个对象的访问权限的特殊情况。禁用 ACL 后，您可以使用策略来控

制对存储桶中所有对象的访问权限，无论是谁将对象上传到您的存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

在此示例中，我们假设存储桶拥有者尚未应用对象所有权的强制存储桶拥有者设置。存储桶拥有者向其账户中的用户委派权限。下面概括介绍演练步骤：



1. 账户 A 管理员用户使用两条语句来附加存储桶策略。
   - 向账户 B 授予跨账户上传对象的权限。
   - 允许用户使用自己的账户访问存储桶中的对象。
2. 账户 B 管理员用户将对象上传至账户 A 拥有的存储桶。
3. 账户 B 管理员将更新对象 ACL，在其中添加授权，以向存储桶拥有者授予对于该对象的完全控制权限。
4. 账户 A 中的用户将通过访问存储桶中的对象（而不管谁拥有它们）来验证权限。

对于此示例，您需要两个账户。下表显示我们如何引用这些账户和这些账户中的管理员用户。根据建议的 IAM 指南，在本演练中，请不要使用账户根用户凭证。有关更多信息，请参阅 关于使用管理员用户来创建资源和授予权限。而是在每个账户中创建一个管理员，在创建资源和向他们授予权限时使用这些凭证。

AWS 账户 ID	账户名称	账户中的管理员
1111-1111-1111	账户 A	AccountAadmin

AWS 账户 ID	账户名称	账户中的管理员
2222-2222-2222	账户 B	AccountBadmin

创建用户和授予权限的所有任务都在 AWS Management Console 中完成。为验证权限，演练中使用命令行工具、AWS Command Line Interface (AWS CLI) 和 AWS Tools for Windows PowerShell，因此您无需编写任何代码。

步骤 0：准备演练

1. 确保您有两个 AWS 账户，并且每个账户都有一个管理员，如前面部分的表中所示。

   a. 根据需要注册 AWS 账户。

   b. 使用账户 A 凭证登录 IAM 控制台，然后执行以下操作以创建管理员用户：

      - 创建用户 **AccountAadmin** 并记下该用户的安全凭证。有关添加用户的更多信息，请参阅 IAM 用户指南中的在您的 AWS 账户中创建 IAM 用户。
      - 通过附加一个授予完全访问权限的用户策略来向 AccountAadmin 授予管理员权限。有关说明，请参阅《IAM 用户指南》中的管理 IAM 策略。
      - 在 IAM 控制台的控制面板中，记下 IAM 用户登录 URL。此账户中的用户必须在登录 AWS Management Console 时使用此 URL。有关更多信息，请参阅 IAM 用户指南中的用户如何登录您的账户。

   c. 使用账户 B 凭证重复上一个步骤，然后创建管理员用户 **AccountBadmin**。

2. 设置 AWS CLI 或 Tools for Windows PowerShell。请务必保存管理员凭证，如下所示：

   - 如果使用 AWS CLI，请在配置文件中创建两个配置文件：`AccountAadmin` 和 `AccountBadmin`。
   - 如果使用 Tools for Windows PowerShell，请确保将用于会话的凭证存储为 `AccountAadmin` 和 `AccountBadmin`。

   有关说明，请参阅 设置用于演练的工具。

步骤 1：执行账户 A 任务

对账户 A 执行以下步骤：

步骤 1.1：登录到控制台

使用账户 A 的 IAM 用户登录 URL，以 **AccountAadmin** 用户身份登录 AWS Management Console。此用户将创建一个存储桶并向其附加一个策略。

步骤 1.2：创建存储桶和用户，并添加一条授予用户权限的存储桶策略

1. 在 Amazon S3 控制台中创建一个存储桶。此练习假设该存储桶在美国东部（弗吉尼亚州北部）AWS 区域创建，名称为 *amzn-s3-demo-bucket1*。

   有关说明，请参阅 创建存储桶。

2. 在 IAM 控制台中，创建用户 **Dave**。

   有关分步说明，请参阅《IAM 用户指南》中的创建 IAM 用户（控制台）。

3. 记下用户 Dave 的凭证。

4. 在 Amazon S3 控制台中，将以下存储桶策略附加到 *amzn-s3-demo-bucket1* 存储桶。有关说明，请参阅 使用 Amazon S3 控制台添加存储桶策略。按照这些步骤添加存储桶策略。有关如何查找账户 ID 的信息，请参阅查找您的 AWS 账户 ID。

   该策略将向账户 B 授予 s3:PutObject 和 s3:ListBucket 权限。该策略还为用户 Dave 授予了 s3:GetObject 权限。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Statement1",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountB-ID:root"
         },
         "Action": [
            "s3:PutObject",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "arn:aws:s3:::amzn-s3-demo-bucket1"
         ]
      },
      {
```

```
            "Sid": "Statement3",
            "Effect": "Allow",
            "Principal": {
               "AWS": "arn:aws:iam::AccountA-ID:user/Dave"
            },
            "Action": [
               "s3:GetObject"
            ],
            "Resource": [
               "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
         }
      ]
}
```

步骤 2：执行账户 B 任务

现在账户 B 有权对账户 A 的存储桶执行操作，账户 B 管理员将执行以下操作：

- 将对象上传到账户 A 的存储桶。
- 在对象 ACL 中添加授权，向账户 A（存储桶拥有者）授予完全控制权限。

使用 AWS CLI

1. 使用 put-object AWS CLI 命令，上传对象。该命令中的 --body 参数指示要上传的源文件。例如，如果该文件在 Windows 计算机上的 C: 驱动器中，您应指定 c:\HappyFace.jpg。--key 参数提供对象的键名。

   ```
   aws s3api put-object --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --body
    HappyFace.jpg --profile AccountBadmin
   ```

2. 在对象 ACL 中添加一个授权，向存储桶拥有者授予对于对象的完全控制权。有关如何查找规范用户 ID 的信息，请参阅《AWS 账户管理参考指南》中的 Find the canonical user ID for your AWS 账户。

   ```
   aws s3api put-object-acl --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --grant-
   full-control id="AccountA-CanonicalUserID" --profile AccountBadmin
   ```

使用 Tools for Windows PowerShell

1. 使用 `Write-S3Object` 命令，上传对象。

```
Write-S3Object -BucketName amzn-s3-demo-bucket1 -key HappyFace.jpg -file
 HappyFace.jpg -StoredCredentials AccountBadmin
```

2. 在对象 ACL 中添加一个授权，向存储桶拥有者授予对于对象的完全控制权。

```
Set-S3ACL -BucketName amzn-s3-demo-bucket1 -Key HappyFace.jpg -CannedACLName
 "bucket-owner-full-control" -StoredCreden
```

步骤 3：测试权限

现在验证账户 A 中的用户 Dave 是否能够访问账户 B 拥有的对象。

使用 AWS CLI

1. 将用户 Dave 的凭证添加到 AWS CLI 配置文件，并创建新的配置文件 `UserDaveAccountA`。有关更多信息，请参阅 设置用于演练的工具。

```
[profile UserDaveAccountA]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. 运行 `get-object` CLI 命令，下载 `HappyFace.jpg`，并将它保存在本地。您可以通过添加参数 `--profile` 为用户 Dave 提供凭证。

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key
 HappyFace.jpg Outputfile.jpg --profile UserDaveAccountA
```

使用 Tools for Windows PowerShell

1. 将用户 Dave 的 AWS 凭证以 `UserDaveAccountA` 的身份保存到持久存储。

```
Set-AWSCredentials -AccessKey UserDave-AccessKey -SecretKey UserDave-
SecretAccessKey -storeas UserDaveAccountA
```

2. 运行 Read-S3Object 命令，下载 HappyFace.jpg 对象，并将它保存在本地。您可以通过添加参数 -StoredCredentials 为用户 Dave 提供凭证。

```
Read-S3Object -BucketName amzn-s3-demo-bucket1 -Key HappyFace.jpg -file
 HappyFace.jpg  -StoredCredentials UserDaveAccountA
```

步骤 4：清除

1. 完成测试之后，您可以执行以下操作来进行清理：
   - 使用账户 A 凭证登录 AWS Management Console，并执行以下操作：
     - 在 Amazon S3 控制台中，移除附加到 *amzn-s3-demo-bucket1* 的存储桶策略。在存储桶 Properties（属性）中，删除 Permissions（权限）部分中的策略。
     - 如果该存储桶是为此练习而创建的，请在 Amazon S3 控制台中删除对象，然后删除存储桶。
     - 在 IAM 控制台中，移除 AccountAadmin 用户。有关分步说明，请参阅《IAM 用户指南》中的删除 IAM 用户。
2. 使用账户 B 凭证登录 AWS Management Console。在 IAM 控制台中，删除用户 AccountBadmin。

示例 4：存储桶拥有者针对自己未拥有的对象授予跨账户权限

主题

在此示例场景中，您拥有一个存储桶，并且启用了其他 AWS 账户以上传对象。如果您已经为存储桶的 S3 对象所有权应用了强制存储桶拥有者设置，那么您将拥有存储桶中的所有对象，包括另一个 AWS 账户 所编写的对象。此方法将解决存储桶拥有者未拥有对象的问题。然后，您可以将权限委派给自己

账户中的用户或其他 AWS 账户。假设尚未启用 S3 对象所有权的强制存储桶拥有者设置。也就是说，您的存储桶中可以有其他 AWS 账户所拥有的对象。

现在，假设作为存储桶拥有者，您需要向另一个账户中的用户授予对象（无论拥有者是谁）的跨账户权限。例如，该用户可以是需要访问对象元数据的账单应用程序。存在两个核心问题：

- 存储桶拥有者对其他 AWS 账户创建的对象不拥有权限。因此，存储桶拥有者若要针对自己未拥有的对象授予权限，必须先由对象拥有者向存储桶拥有者授予权限。对象拥有者是创建对象版本的 AWS 账户。然后，存储桶拥有者才能委托这些权限。
- 存储桶拥有者账户可以向其账户中的用户委派权限（请参阅示例 3：存储桶拥有者授予不属于自己的对象的权限）。但是，存储桶拥有者账户无法将权限委派给其它 AWS 账户，因为不支持跨账户委派。

在这种情况下，存储桶拥有者可以创建具有对象访问权限的 AWS Identity and Access Management（IAM）角色，然后存储桶拥有者可以向另一个 AWS 账户授予临时担任该角色的权限，使它可以访问存储桶中的对象。

> **Note**
>
> S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权和禁用或启用 ACL。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对这些对象的访问权限。
>
> Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有需要单独控制每个对象的访问权限的特殊情况。禁用 ACL 后，您可以使用策略来控制对存储桶中所有对象的访问权限，无论是谁将对象上传到您的存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

了解跨账户权限和使用 IAM 角色

IAM 角色可实现多种资源委托访问权限方案，跨账户访问是重要方案之一。在此示例中，存储桶拥有者（账户 A）使用一个 IAM 角色临时将跨账户对象访问权限委派给另一个 AWS 账户（账户 C）中的用户。您创建的每个 IAM 角色都附加了以下两个策略：

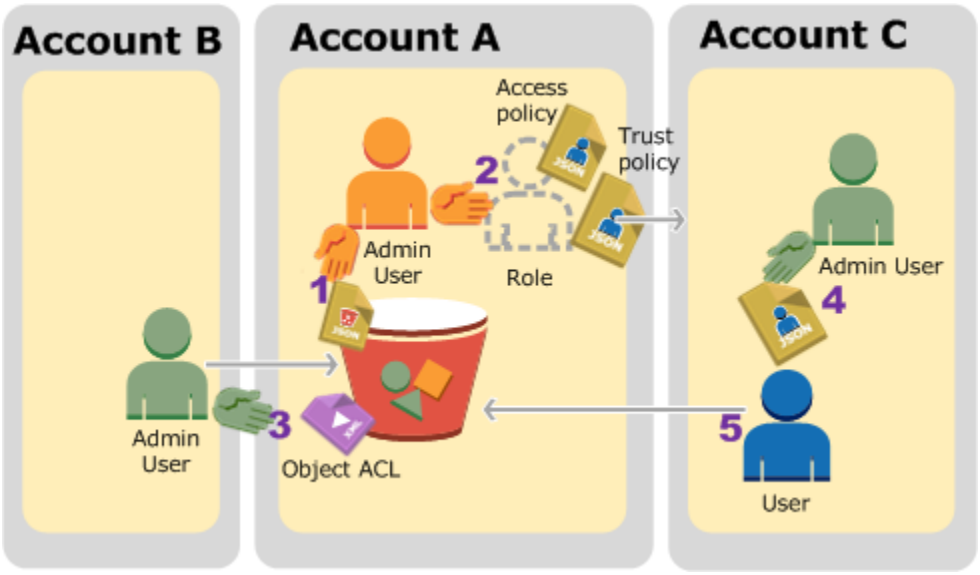
- 一个信任策略，用于标识可以担任角色的其他 AWS 账户。
- 一个访问策略，用于定义在某个用户担任角色时允许的权限，例如 `s3:GetObject`。有关可以在策略中指定的权限的列表，请参阅 Amazon S3 的策略操作。

然后，信任策略中标识的 AWS 账户向其用户授予担任角色的权限。该用户即可执行以下操作以访问对象：

- 担任角色，然后在响应中获取临时安全凭证。
- 使用临时安全凭证访问存储桶中的对象。

有关 IAM 角色的更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

下面概括介绍演练步骤：



1. 账户 A 管理员用户附加一个存储桶策略，向账户 B 授予上传对象的条件权限。
2. 账户 A 管理员创建一个 IAM 角色，与账户 C 建立信任，以便该账户中的用户可以访问账户 A。附加到该角色的访问策略限制账户 C 中的用户访问账户 A 时可以执行的操作。
3. 账户 B 管理员将一个对象上传到账户 A 拥有的存储桶，从而向存储桶拥有者授予完全控制权限。
4. 账户 C 管理员创建一个用户，附加一个用户策略允许该用户担任该角色。
5. 账户 C 中的用户首先担任角色，这会向该用户返回临时安全凭证。该用户随后使用这些临时凭证访问存储桶中的对象。

对于此示例，您需要三个账户。下表显示我们如何引用这些账户和这些账户中的管理员用户。根据 IAM 准则（请参阅关于使用管理员用户来创建资源和授予权限），我们在本演练中并不使用 AWS 账户根用户 凭证。而是在每个账户中创建一个管理员用户，并在创建资源和向他们授予权限时使用这些凭证。

AWS 账户 ID	账户名称	账户中的管理员用户
1111-1111-1111	账户 A	AccountAadmin
2222-2222-2222	账户 B	AccountBadmin
3333-3333-3333	账户 C	AccountCadmin

步骤 0：准备演练

> **Note**
>
> 在演练这些步骤时，您可能需要打开文本编辑器，记下某些信息。具体而言，您需要账户 ID、规范用户 ID、供每个账户连接到控制台的 IAM 用户登录 URL，以及 IAM 用户和角色的 Amazon 资源名称（ARN）。

1. 确保您有三个 AWS 账户并且每个账户都有一个管理员用户，如前面部分的表中所示。

   a. 根据需要注册 AWS 账户。我们将这些账户称为账户 A、账户 B 和账户 C。

   b. 使用账户 A 凭证登录 IAM 控制台，然后执行以下操作以创建管理员用户：

   - 创建用户 **`AccountAadmin`** 并记下其安全凭证。有关添加用户的更多信息，请参阅 IAM 用户指南中的在您的 AWS 账户中创建 IAM 用户。
   - 通过附加一个授予完全访问权限的用户策略来向 AccountAadmin 授予管理员权限。有关说明，请参阅《IAM 用户指南》中的管理 IAM 策略。
   - 在 IAM 控制台的控制面板中，记下 IAM 用户登录 URL。此账户中的用户必须在登录 AWS Management Console 时使用此 URL。有关更多信息，请参阅《IAM 用户指南》中的 Sign in to the AWS Management Console as an IAM user。

   c. 重复上述步骤，在账户 B 和账户 C 中创建管理员用户。

2. 对于账户 C，请记下规范用户 ID。

   在账户 A 中创建 IAM 角色时，信任策略通过指定账户 ID 向账户 C 授予担任该角色的权限。您可以查找账户信息，如下所示：

   a. 使用 AWS 账户 ID 或账户别名、您的 IAM 用户名和密码登录 Amazon S3 控制台。

b. 请选择 Amazon S3 存储桶的名称以查看有关该存储桶的详细信息。

c. 请选择 Permissions (权限) 选项卡，然后选择 Access Control List (访问控制列表)。

d. 在 Access for your AWS 账户 (您的 Amazon 账户的访问权限) 部分，Account (账户) 列中是长标识符，例如 `c1daexampleaaf850ea79cf0430f33d72579fd1611c97f7ded193374c0b163b6`。这是您的规范用户 ID。

3. 创建存储桶策略时，您需要以下信息。记下以下值：

- 账户 A 的规范用户 ID – 账户 A 管理员向账户 B 管理员授予条件上传对象权限时，条件指定必须获取对象完全控制权的账户 A 用户的规范用户 ID。

> **Note**
>
> 规范用户 ID 是 Amazon S3 独有的概念。它是 64 字符模糊版本的账户 ID。

- 账户 B 管理员的用户 ARN：您可以在 IAM 控制台中找到用户 ARN。您必须选择该用户并在摘要选项卡中找到该用户的 ARN。

  在存储桶策略中，向 `AccountBadmin` 授予上传对象的权限，使用 ARN 指定用户。下面是一个示例 ARN 值：

```
arn:aws:iam::AccountB-ID:user/AccountBadmin
```

4. 设置 AWS Command Line Interface (CLI) 或 AWS Tools for Windows PowerShell。请务必保存管理员用户凭证，如下所示：

- 如果使用 AWS CLI，请在配置文件中创建配置文件：`AccountAadmin` 和 `AccountBadmin`。
- 如果使用 AWS Tools for Windows PowerShell，请确保将用于会话的凭证存储为 `AccountAadmin` 和 `AccountBadmin`。

  有关说明，请参阅 设置用于演练的工具。

步骤 1：执行账户 A 任务

在此示例中，账户 A 是存储桶拥有者。因此，账户 A 中的用户 AccountAadmin 将执行以下操作：

- 创建存储桶。

- 附加存储桶策略，向账户 B 管理员授予上传对象的权限。
- 创建 IAM 角色，授予账户 C 代入该角色的权限，使其能够访问存储桶中的对象。

步骤 1.1：登录到 AWS Management Console

首先使用账户 A 的 IAM 用户登录 URL，以 **AccountAadmin** 用户身份登录 AWS Management Console。此用户将创建一个存储桶并向其附加一个策略。

步骤 1.2：创建存储桶并附加存储桶策略

在 Amazon S3 控制台中，执行以下操作：

1. 创建存储桶。此练习假设存储桶名称是 *amzn-s3-demo-bucket1*。

   有关说明，请参阅 创建存储桶。
2. 附加以下存储桶策略。该策略向账户 B 管理员授予上传对象的条件权限。

   请通过为 *amzn-s3-demo-bucket1*、*AccountB-ID* 和 *CanonicalUserId-of-AWSaccountA-BucketOwner* 提供自己的值来更新策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "111",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        },
        {
            "Sid": "112",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "StringNotEquals": {
```

```
                    "s3:x-amz-grant-full-control": "id=CanonicalUserId-of-
AWSaccountA-BucketOwner"
                }
            }
        }
    ]
}
```

步骤 1.3：在账户 A 中创建一个 IAM 角色以允许账户 C 进行跨账户访问

在 IAM 控制台中创建一个 IAM 角色（**examplerole**），授予账户 C 担任该角色的权限。请务必仍以账户 A 管理员身份登录，因为该角色必须在账户 A 中创建。

1. 创建角色之前，准备好定义角色所需权限的托管策略。您可在以后的步骤中将此策略附加到角色。

   a. 在左侧导航窗格中，请选择策略，然后选择创建策略。

   b. 在 Create Your Own Policy 旁，选择 Select。

   c. 在 Policy Name (策略名称) 字段中输入 **access-accountA-bucket**。

   d. 复制下面的访问策略，然后将其粘贴到 Policy Document (策略文档) 字段中。该访问策略授予该角色 `s3:GetObject` 权限，这样，账户 C 用户在担任该角色时只能执行 `s3:GetObject` 操作。

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
       }
     ]
   }
   ```

   e. 选择创建策略。

   新策略会显示在托管策略列表中。

2. 在左侧的导航窗格中，选择角色，然后选择创建角色。

3. 在选择角色类型下，选择用于跨账户访问的角色，然后单击提供在您的 AWS 账户之间进行相互访问的权限旁边的选择按钮。

4. 输入账户 C 的账户 ID。

   对于此演练，您不必要求用户经过多重身份验证（MFA）来承担角色，因此请不要选择该选项。

5. 选择下一步，设置与该角色关联的权限。

6. 选中您创建的 access-accountA-bucket 策略旁边的复选框，然后选择下一步。

   “Review (审核)”页面随即出现，使您可以在创建角色之前确认其设置。此页面上有一项极为重要，需要注意，即您可发送给需要使用此角色的用户的链接。使用该链接的用户会直接转到已填写“账户 ID”和“角色名称 ”字段的切换角色页面。稍后您可以在任何跨账户角色的角色摘要页面上看到此链接。

7. 输入 `examplerole` 作为角色名称，然后单击下一步。

8. 检查角色后，选择创建角色。

   `examplerole` 角色显示在角色列表中。

9. 选择角色名称 `examplerole`。

10. 选择 Trust Relationships (信任关系) 选项卡。

11. 选择显示策略文档并验证所示信任策略是否与以下策略相符。

    以下信任策略通过允许账户 C 执行 `sts:AssumeRole` 操作，与它建立信任。有关更多信息，请参阅《AWS Security Token Service API 参考》中的 AssumeRole。

    ```
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Sid": "",
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::AccountC-ID:root"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
    ```

12. 记下您创建的 `examplerole` 角色的 Amazon 资源名称（ARN）。

    稍后在以下步骤中，您附加一个用户策略以允许 IAM 用户担任此角色，并通过 ARN 值标识角色。

步骤 2：执行账户 B 任务

账户 A 拥有的示例存储桶需要由其它账户拥有的对象。在此步骤中，账户 B 管理员使用命令行工具上传对象。

- 使用 put-object AWS CLI 命令，将对象上传到 *amzn-s3-demo-bucket1*。

```
aws s3api put-object --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --
body HappyFace.jpg --grant-full-control id="canonicalUserId-ofTheBucketOwner" --
profile AccountBadmin
```

请注意以下几点：

- `--Profile` 参数指定 `AccountBadmin` 配置文件，因此该对象由账户 B 拥有。
- 参数 `grant-full-control` 根据存储桶策略的要求向存储桶拥有者授予对象的完全控制权限。
- `--body` 参数标识要上传的源文件。例如，如果该文件在 Windows 计算机上的 C: 驱动器中，您应指定 `c:\HappyFace.jpg`。

步骤 3：执行账户 C 任务

在前面的步骤中，账户 A 已创建一个角色 `examplerole`，与账户 C 建立了信任。该角色使账户 C 中的用户可以访问账户 A。在此步骤中，账户 C 管理员创建一个用户（Dave），向他委派从账户 A 收到的 `sts:AssumeRole` 权限。此方法让 Dave 可以担任 `examplerole` 并临时获取对账户 A 的访问权限。账户 A 附加到该角色的访问策略将限制 Dave 在访问账户 A 时可以执行的操作，具体而言，就是获取 *amzn-s3-demo-bucket1* 中的对象。

步骤 3.1：在账户 C 中创建一个用户并向其委派担任 examplerole 的权限

1. 首先使用账户 C 的 IAM 用户登录 URL，以 **`AccountCadmin`** 用户身份登录 AWS Management Console。

2. 在 IAM 控制台中，创建用户 Dave。

   有关分步说明，请参阅《IAM 用户指南》中的创建 IAM 用户（AWS Management Console）。

3. 记下 Dave 的凭证。Dave 需要这些凭证才能担任 `examplerole` 角色。
4. 为 IAM 用户 Dave 创建一个内联策略，向 Dave 委派担任账户 A 中的 `examplerole` 角色的 `sts:AssumeRole` 权限。

a. 在左侧导航窗格中，请选择用户。

b. 请选择用户名 Dave。

c. 在用户详细信息页面上，选择 Permissions (权限) 选项卡，然后展开 Inline Policies (内联策略) 部分。

d. 请选择 click here (单击此处) (或 Create User Policy (创建用户策略))。

e. 选择 Custom Policy，然后选择 Select。

f. 在 Policy Name (策略名称) 字段中为策略输入一个名称。

g. 将以下策略复制到 Policy Document (策略文档) 字段中。

   您必须提供 *AccountA-ID* 来更新策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole"],
      "Resource": "arn:aws:iam::AccountA-ID:role/examplerole"
    }
  ]
}
```

h. 选择应用策略。

5. 通过添加另一个配置文件 AccountCDave 将 Dave 的凭证保存到 AWS CLI 的配置文件。

```
[profile AccountCDave]
aws_access_key_id = UserDaveAccessKeyID
aws_secret_access_key = UserDaveSecretAccessKey
region = us-west-2
```

步骤 3.2：担任角色（examplerole）并访问对象

现在 Dave 可以访问账户 A 拥有的存储桶中的对象，如下所示：

- Dave 首先使用自己的凭证担任 examplerole。这会返回临时凭证。
- Dave 随后使用临时凭证访问账户 A 的存储桶中的对象。

1. 在命令提示符处，使用 AccountCDave 配置文件运行下面的 AWS CLI assume-role 命令。

   您必须通过提供 *AccountA-ID*（其中定义了 examplerole）在命令中更新 ARN 值。

   ```
   aws sts assume-role --role-arn arn:aws:iam::AccountA-ID:role/examplerole --profile
    AccountCDave --role-session-name test
   ```

   在响应中，AWS Security Token Service（AWS STS）返回临时安全凭证（访问密钥 ID、秘密访问密钥和会话令牌）。

2. 将临时安全凭证保存在 AWS CLI 配置文件中的 TempCred 配置文件下。

   ```
   [profile TempCred]
   aws_access_key_id = temp-access-key-ID
   aws_secret_access_key = temp-secret-access-key
   aws_session_token = session-token
   region = us-west-2
   ```

3. 在命令提示符处，使用临时凭证运行以下 AWS CLI 命令以访问对象。例如，该命令指定 head-object API 以检索 HappyFace.jpg 对象的对象元数据。

   ```
   aws s3api get-object --bucket amzn-s3-demo-bucket1 --
   key HappyFace.jpg SaveFileAs.jpg --profile TempCred
   ```

   因为附加到 examplerole 的访问策略允许执行这些操作，所以 Amazon S3 会处理请求。您可以对存储桶中的任何其他对象尝试任何其他操作。

   如果您尝试任何其它操作（例如 get-object-acl），则系统会拒绝权限，因为不允许角色执行该操作。

   ```
   aws s3api get-object-acl --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --
   profile TempCred
   ```

   我们使用用户 Dave 担任角色，并使用临时凭证访问对象。它还可以是账户 C 中访问 *amzn-s3-demo-bucket1* 中的对象的应用程序。应用程序可以获取临时安全凭证，账户 C 可以向应用程序委托担任 examplerole 的权限。

步骤 4：清除

1. 完成测试之后，您可以执行以下操作来进行清理：

- 使用账户 A 凭证登录 AWS Management Console，并执行以下操作：
  - 在 Amazon S3 控制台中，移除附加到 *amzn-s3-demo-bucket1* 的存储桶策略。在存储桶 Properties（属性）中，删除 Permissions（权限）部分中的策略。
  - 如果该存储桶是为此练习而创建的，请在 Amazon S3 控制台中删除对象，然后删除存储桶。
  - 在 IAM 控制台中，删除您在账户 A 中创建的 `examplerole`。有关分步说明，请参阅《IAM 用户指南》中的删除 IAM 用户。
  - 在 IAM 控制台中，移除 AccountAadmin 用户。

2. 使用账户 B 凭证登录 IAM 控制台。删除用户 AccountBadmin。
3. 使用账户 C 凭证登录 IAM 控制台。删除 AccountCadmin 和用户 Dave。

相关资源

有关本演练的更多信息，请参阅《IAM 用户指南》中的以下资源：

- 创建向 IAM 用户委派权限的角色
- 教程：使用 IAM 角色委派跨 AWS 账户的访问权限
- 管理 IAM 策略

Amazon S3 如何对请求授权

当 Amazon S3 收到请求（例如，存储桶或对象操作）时，它首先验证请求者是否拥有必要的权限。Amazon S3 对所有相关访问策略、用户策略和基于资源的策略（存储桶策略、存储桶访问控制列表（ACL）、对象 ACL）进行评估，以决定是否对该请求进行授权。

> **Note**
>
> 如果 Amazon S3 权限检查未能找到有效的权限，将返回拒绝访问（403 禁止）错误。有关更多信息，请参阅排查 Amazon S3 中的拒绝访问（403 禁止）错误。

为了确定请求者是否拥有执行特定操作的权限，Amazon S3 会在收到请求时按顺序执行以下操作：

1. 在运行时将所有相关访问策略（用户策略、存储桶策略、ACL）转换为一组策略以进行评估。
2. 通过以下步骤评估生成的策略集。在每个步骤中，Amazon S3 都会基于特定上下文机构来评估上下文中的策略子集。
   a. 用户上下文 – 在用户上下文中，用户所属的父账户是上下文机构。

      Amazon S3 对父账户拥有的策略子集进行评估。该子集包括父账户附加到该用户的用户策略。如果父账户也拥有请求中的资源（存储桶或对象），则 Amazon S3 还会同时评估相应资源策略（存储桶策略、存储桶 ACL 和对象 ACL）。

      用户必须从父账户获得执行该操作的权限。

      只有在 AWS 账户中的用户发出请求的情况下，此步骤才适用。如果请求是使用 AWS 账户 的根用户凭证发出的，则 Amazon S3 跳过此步骤。

   b. 存储桶上下文 – 在存储桶上下文中，Amazon S3 评估拥有该存储桶的 AWS 账户 所拥有的策略。

      如果请求是针对存储桶操作发出的，则请求者必须拥有来自存储桶拥有者的权限。如果请求是针对对象发出的，则 Amazon S3 会评估由存储桶拥有者拥有的所有策略，以检查存储桶拥有者是否未显式拒绝对该对象的访问。如果设置了显式拒绝，则 Amazon S3 不对请求授权。

   c. 对象上下文 – 如果请求是针对对象发出的，则 Amazon S3 对由对象拥有者拥有的策略子集进行评估。

以下是说明 Amazon S3 如何授权请求的一些示例场景。

Example – 请求者是 IAM 主体

如果请求者是 IAM 主体，则 Amazon S3 必须确定主体所属的父 AWS 账户是否为主体授予了所需的权限以执行该操作。此外，如果请求是要执行存储桶操作（例如，请求列出存储桶内容），则 Amazon S3 必须验证存储桶拥有者是否已为请求者授予执行该操作的权限。要对资源执行特定的操作，IAM 主体需要来自所属的父 AWS 账户以及拥有该资源的 AWS 账户的权限。

Example – 请求者是 IAM 主体 - 如果请求针对的是存储桶拥有者未拥有的对象的操作。

如果请求是要针对存储桶拥有者未拥有的对象执行某一操作，则除了确保请求者拥有来自对象拥有者的权限外，Amazon S3 还必须检查存储桶策略，以确保存储桶拥有者未对该对象设置显式拒绝。存储桶拥有者（支付账单者）可以显式拒绝对存储桶中的对象的访问，而不论谁拥有该对象。存储桶拥有者还可以删除存储桶中的任何对象。

默认情况下，当另一个 AWS 账户将对象上传到您的 S3 存储桶时，该账户（对象编写者）拥有该对象，拥有此对象的访问权限，并可以通过访问控制列表（ACL）授予其他用户访问该对象的权限。您可以使用对象所有权来更改此默认行为，以便禁用 ACL，并且作为存储桶拥有者，您可以自动拥有存储桶中的每个对象。因此，数据的访问控制基于策略，例如 IAM 用户策略、S3 存储桶策略、虚拟私有云（VPC）端点策略和 AWS Organizations 服务控制策略（SCP）。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

有关 Amazon S3 如何评估访问策略以授权或拒绝存储桶操作和对象操作请求的更多信息，请参阅以下主题：

主题

- Amazon S3 如何对存储桶操作请求进行授权
- Amazon S3 如何授权对象操作的请求

Amazon S3 如何对存储桶操作请求进行授权

当 Amazon S3 收到存储桶操作请求时，Amazon S3 会将所有相关的权限转换为一组策略，以在运行时评估。相关权限包括基于资源的权限（例如，存储桶策略和存储桶访问控制列表）和用户策略（如果请求来自 IAM 主体）。然后，Amazon S3 将通过一系列步骤根据特定上下文（用户上下文或存储桶上下文）来评估生成的策略集：

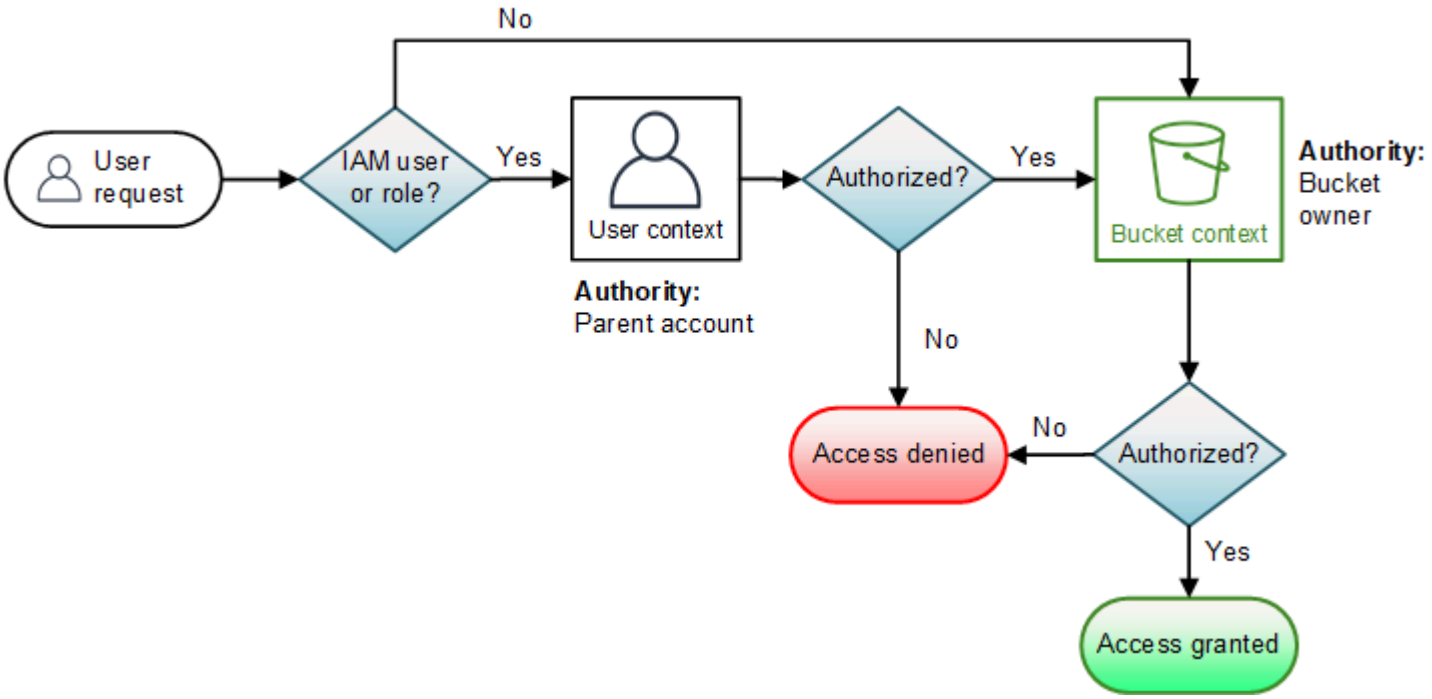
1. 用户上下文 – 如果请求者是 IAM 主体，则主体必须具有来自所属的父 AWS 账户 的权限。在此步骤中，Amazon S3 将评估由父账户 (也称为上下文机构) 拥有的一个策略子集。该策略子集包括父账户附加到主体的用户策略。如果父账户也拥有请求中的资源（在本例中为存储桶），则 Amazon S3 还

会同时评估相应资源策略（存储桶策略和存储桶 ACL）。无论何时发出存储桶操作请求，服务器访问日志都会记录请求者的规范 ID。有关更多信息，请参阅 使用服务器访问日志记录来记录请求。

2. 存储桶上下文 – 请求者必须拥有来自存储桶拥有者的权限才能执行特定存储桶操作。在此步骤中，Amazon S3 对由拥有该存储桶的 AWS 账户拥有的策略子集进行评估。

   存储桶拥有者可通过使用存储桶策略或存储桶 ACL 来授予权限。如果拥有存储桶的 AWS 账户也是 IAM 主体的父账户，则该账户可以在用户策略中配置存储桶权限。

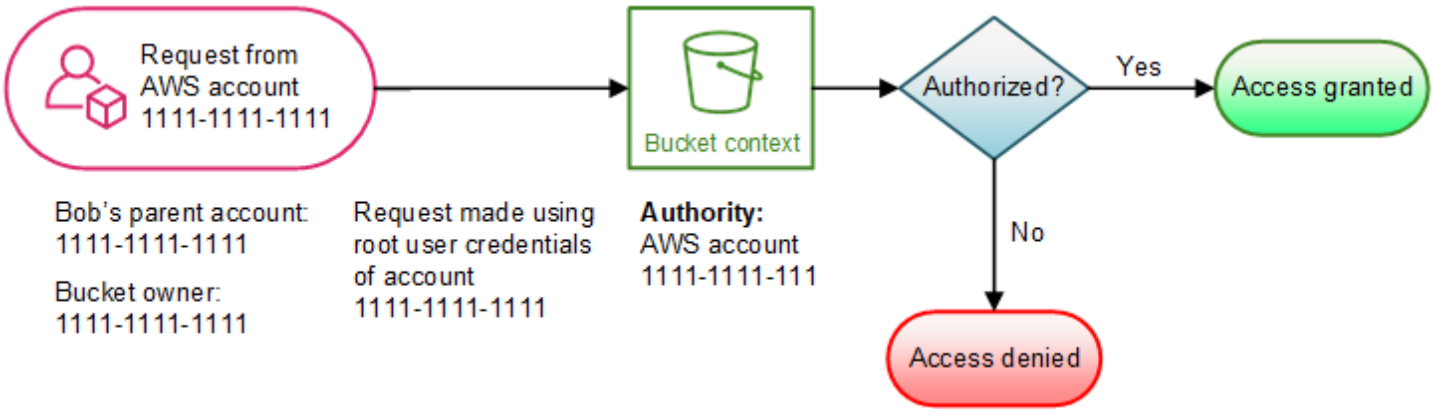
下面是基于上下文对存储桶操作进行评估的图解说明。



下面的示例演示了评估逻辑。

示例 1：由存储桶拥有者请求的存储桶操作

在此示例中，存储桶拥有者使用 AWS 账户 的根用户凭证发送存储桶操作请求。

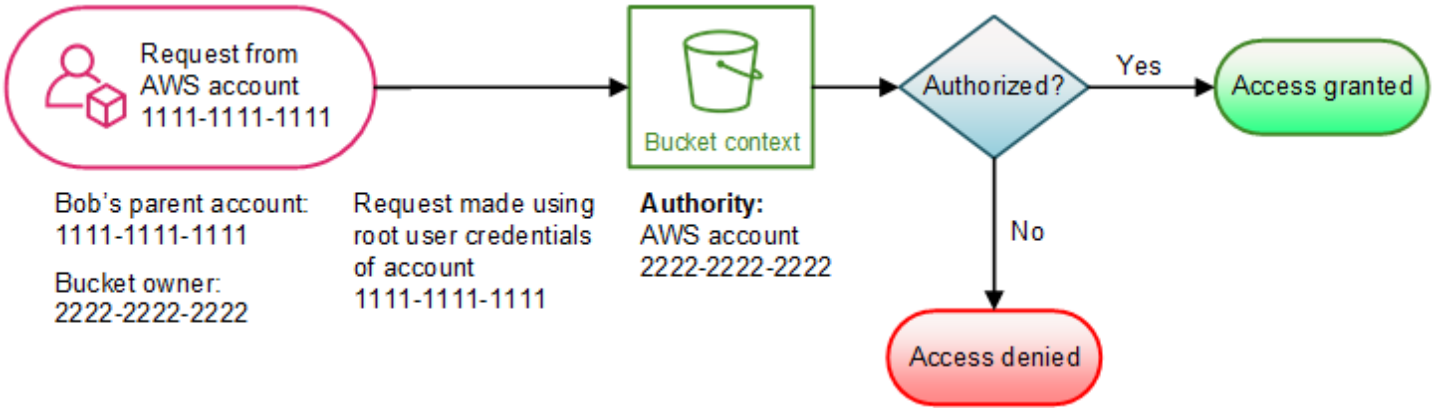


Amazon S3 通过以下方式执行上下文评估：

1. 由于请求是通过使用 AWS 账户 的根用户凭证发出的，因此不评估用户上下文。
2. 在存储桶上下文中，Amazon S3 检查存储桶策略以确定请求者是否拥有执行该操作的权限。Amazon S3 对该请求进行授权。

示例 2：由不是存储桶拥有者的 AWS 账户请求的存储桶操作

在此示例中，使用 AWS 账户 1111-1111-1111 的根用户凭证发出请求，请求执行由 AWS 账户 2222-2222-2222 拥有的存储桶操作。该请求不涉及任何 IAM 用户。

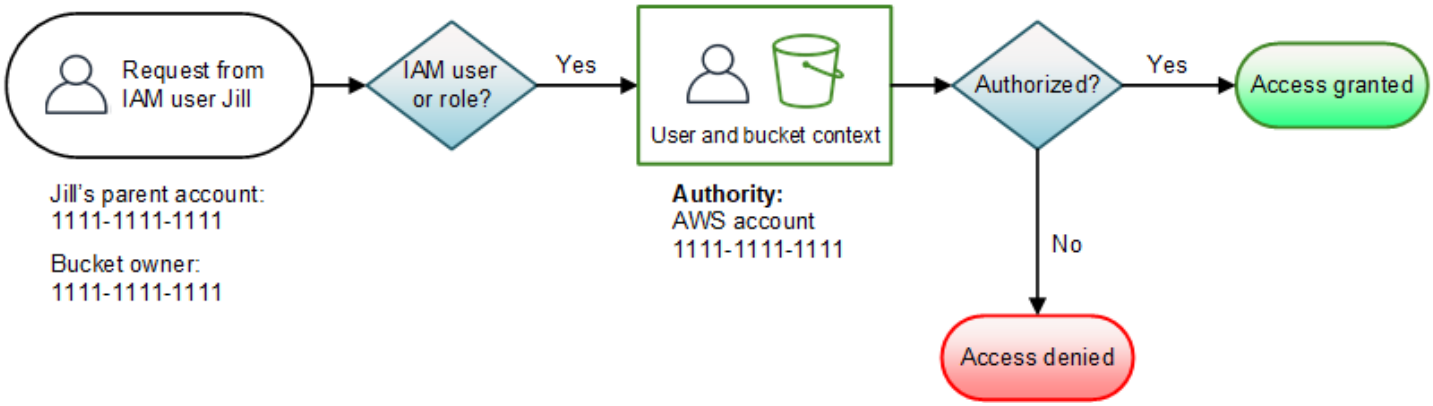


在此示例中，Amazon S3 通过以下方式评估上下文：

1. 由于请求是通过使用 AWS 账户 的根用户凭证发出的，因此不评估用户上下文。
2. 在存储桶上下文中，Amazon S3 检查存储桶策略。如果存储桶拥有者（AWS 账户 2222-2222-2222）尚未授权 AWS 账户 1111-1111-1111 执行请求的操作，则 Amazon S3 拒绝该请求。否则，Amazon S3 授权该请求并执行该操作。

示例 3：IAM 主体请求的存储桶操作，其父 AWS 账户也是存储桶拥有者

在此示例中，请求由 AWS 账户 1111-1111-1111 中的 IAM 用户 Jill 发送，该账户同时也拥有该存储桶。



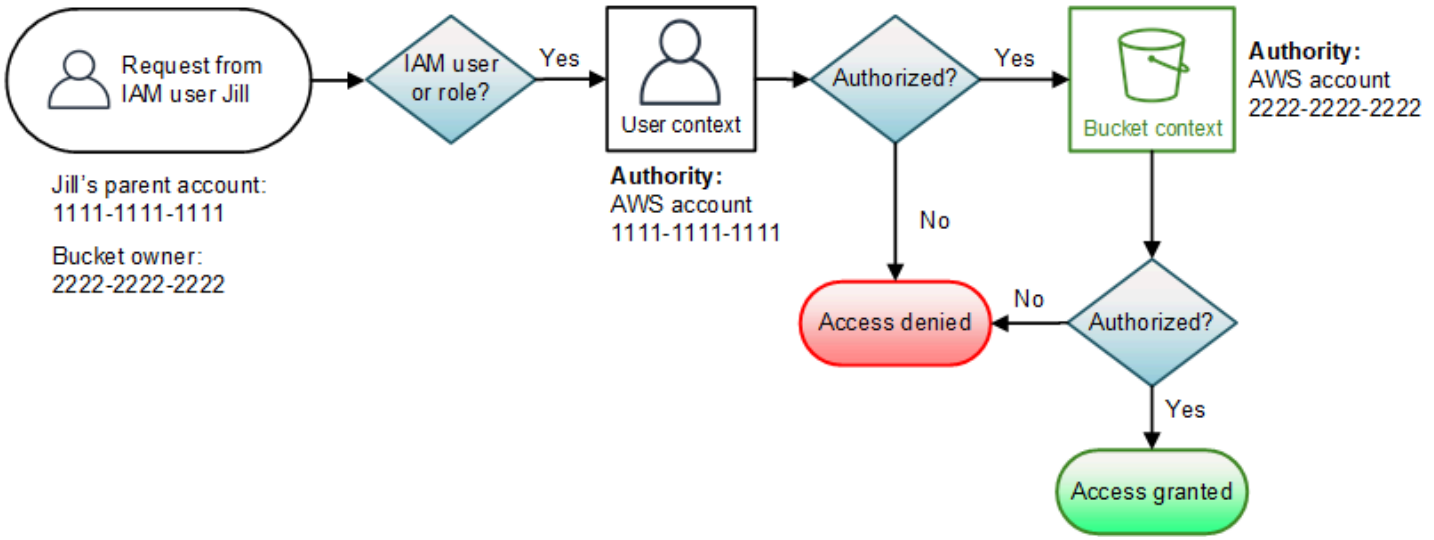
Amazon S3 执行以下上下文评估：

1. 由于请求来自 IAM 主体，因此，在用户上下文中，Amazon S3 评估属于父 AWS 账户的所有策略以确定 Jill 是否有权执行该操作。

   在该示例中，主体所属的父 AWS 账户 1111-1111-1111 也是存储桶拥有者。因此，除用户策略外，Amazon S3 还要评估同一上下文中的存储桶策略和存储桶 ACL，因为它们属于同一账户。
2. 由于 Amazon S3 已将存储桶策略和存储桶 ACL 作为用户上下文的一部分进行了评估，因此它不会评估存储桶上下文。

示例 4：IAM 主体请求的存储桶操作，其父 AWS 账户不是存储桶拥有者

在此示例中，请求是由父 AWS 账户是 1111-1111-1111 的 IAM 用户 Jill 发送的，但存储桶由另一个 AWS 账户 2222-2222-2222 拥有。



Jill 将需要来自父 AWS 账户和存储桶拥有者双方的权限。Amazon S3 通过以下方式评估上下文：

1. 由于请求来自 IAM 主体，因此，Amazon S3 查看账户编写的策略以验证 Jill 是否具有所需的权限，从而对用户上下文进行评估。如果 Jill 拥有权限，则 Amazon S3 会进一步评估存储桶上下文。如果 Jill 没有权限，则会拒绝该请求。
2. 在存储桶上下文中，Amazon S3 验证存储桶拥有者 2222-2222-2222 是否已授予 Jill（或她的父 AWS 账户）执行所请求的操作的权限。如果其拥有该权限，则 Amazon S3 会授权该请求并执行该操作。否则，Amazon S3 拒绝该请求。

Amazon S3 如何授权对象操作的请求

当 Amazon S3 接收对象操作的请求时，它会将所有相关权限转换为一组策略在运行时进行评估，这些相关权限包括：基于资源的权限（对象访问控制列表（ACL）、存储桶策略、存储桶 ACL）和 IAM 用户策略。然后它会通过一系列步骤评估生成的策略集。在每个步骤中，它会在三个特定上下文（用户上下文、存储桶上下文和对象上下文）中评估一个策略子集：

1. 用户上下文 – 如果请求者是 IAM 主体，则主体必须具有来自所属的父 AWS 账户 的权限。在此步骤中，Amazon S3 会评估由父账户 (也称为上下文机构) 拥有的一个策略子集。该策略子集包括父账户附加到主体的用户策略。如果父账户也拥有请求中的资源（存储桶或对象），则 Amazon S3 会同时评估相应资源策略（存储桶策略、存储桶 ACL 和对象 ACL）。

> **Note**
>
> 如果父 AWS 账户拥有资源（存储桶或对象），则该账户可以使用用户策略或资源策略为其 IAM 主体授予资源权限。

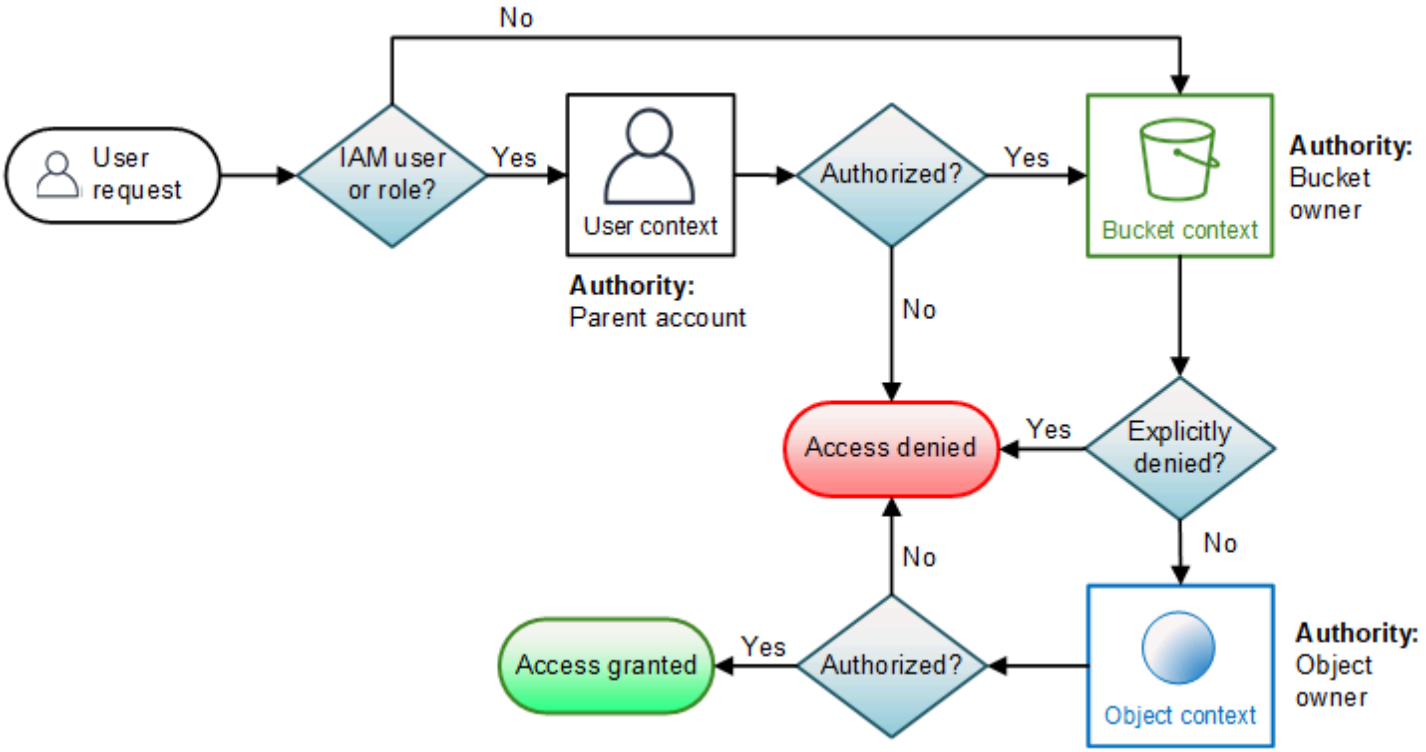
2. 存储桶上下文 – 在此上下文中，Amazon S3 评估拥有该存储桶的 AWS 账户所拥有的策略。

   如果拥有请求中的对象的 AWS 账户与存储桶拥有者不同，则 Amazon S3 会检查策略，查看存储桶拥有者是否已显式拒绝对该对象的访问。如果对该对象设置了显式拒绝，则 Amazon S3 不对请求授权。

3. 对象上下文 – 请求者必须拥有来自对象拥有者的权限才能执行特定对象操作。在此步骤中，Amazon S3 将评估对象 ACL。

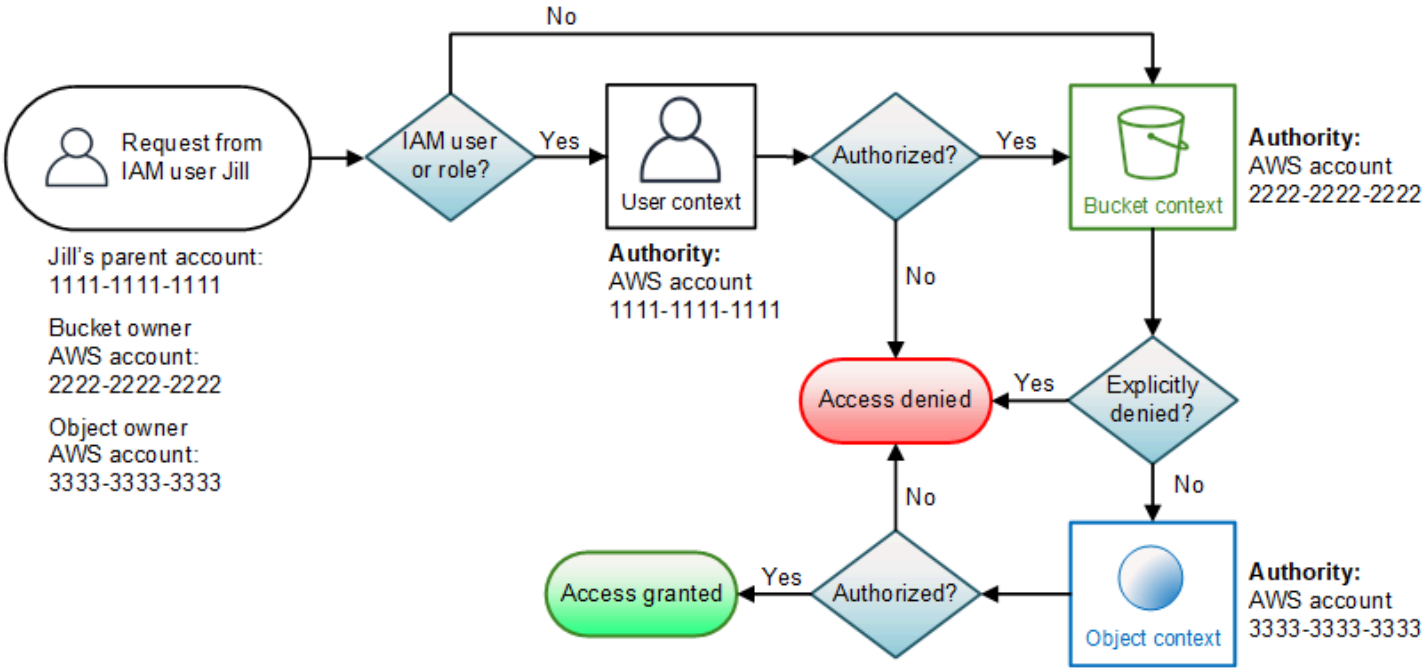
> **Note**
>
> 如果存储桶和对象拥有者相同，则可以在存储桶策略中授予对该对象的访问权限，并会在存储桶上下文中对此进行评估。如果拥有者不同，则对象拥有者必须使用对象 ACL 授予权限。如果拥有对象的 AWS 账户 也是 IAM 主体所属的父账户，则该账户可以在用户策略中配置对象权限，将在用户上下文中对其进行评估。有关使用这些备选访问策略的更多信息，请参阅 演练：使用策略管理针对 Amazon S3 资源的访问权限。
>
> 如果您作为存储桶拥有者想拥有存储桶中的所有对象，并使用存储桶策略或基于 IAM 的策略来管理对这些对象的访问，则可以应用对象所有权的强制存储桶拥有者设置。使用此设置，您作为存储桶拥有者自动拥有并完全控制存储桶中的每个对象。无法编辑存储桶和对象 ACL，也不再考虑访问。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

下面是基于上下文来评估对象操作的图解说明。



对象操作请求示例

在此示例中，IAM 用户 Jill（其父 AWS 账户是 1111-1111-1111）针对 AWS 账户 3333-3333-3333 拥有的对象发送对象操作请求（例如，`GetObject`），该对象位于由 AWS 账户 2222-2222-2222 拥有的存储桶中。



Jill 需要从父 AWS 账户、存储桶拥有者和对象拥有者获得权限。Amazon S3 通过以下方式评估上下文：

1. 由于请求来自 IAM 主体，因此，Amazon S3 评估用户上下文以验证父 AWS 账户 1111-1111-1111 是否为 Jill 授予了权限以执行请求的操作。如果她拥有该权限，则 Amazon S3 评估存储桶上下文。否则，Amazon S3 拒绝该请求。
2. 在存储桶上下文中，存储桶拥有者（AWS 账户 2222-2222-2222）是上下文机构。Amazon S3 对存储桶策略进行评估以确定存储桶拥有者是否显式拒绝了 Jill 对该对象的访问。
3. 在对象上下文中，上下文机构是 AWS 账户 3333-3333-3333，即对象拥有者。Amazon S3 评估对象 ACL 以确定 Jill 是否拥有访问该对象的权限。如果她拥有该权限，则 Amazon S3 对该请求进行授权。

AWS 适用于 Amazon S3 的托管策略

AWS 托管式策略是由 AWS 创建和管理的独立策略。AWS 托管式策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住，AWS 托管式策略可能不会为您的特定使用场景授予最低权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户托管式策略来进一步减少权限。

您无法更改 AWS 托管式策略中定义的权限。如果 AWS 更新在 AWS 托管式策略中定义的权限，则更新会影响该策略所附加到的所有主体身份（用户、组和角色）。当新的 AWS 服务 启动或新的 API 操作可用于现有服务时，AWS 最有可能更新 AWS 托管式策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管策略：AmazonS3FullAccess

您可以将 `AmazonS3FullAccess` 策略附加到 IAM 身份。 此策略授予允许完全访问 Amazon S3 的权限 。

要查看此策略的权限，请参阅 AWS Management Console 中的 AmazonS3FullAccess。

AWS 托管策略：AmazonS3ReadOnlyAccess

您可以将 `AmazonS3ReadOnlyAccess` 策略附加到 IAM 身份。 此策略授予允许对 Amazon S3 进行只读访问的权限。

要查看此策略的权限，请参阅 AWS Management Console 中的 AmazonS3ReadOnlyAccess。

AWS 托管式策略：AmazonS3ObjectLambdaExecutionRolePolicy

提供的 AWS Lambda 函数在向 S3 对象 Lambda 接入点发出请求时将数据发送到 S3 对象 Lambda 所需的权限。还授予 Lambda 写入 Amazon CloudWatch Logs 的权限。

要查看此策略的权限，请参阅 AWS Management Console 中的 AmazonS3ObjectLambdaExecutionRolePolicy。

AWS 托管策略的 Amazon S3 更新

查看有关 Amazon S3 的 AWS 托管策略的更新的详细信息。

更改	描述	日期
Amazon S3 向 AmazonS3ReadOnlyAccess 添加了描述权限	Amazon S3 向 AmazonS3ReadOnlyAccess 添加了 s3:Describe* 权限。	2023 年 8 月 11 日
Amazon S3 将 S3 对象 Lambda 权限添加到 AmazonS3FullAccess 和 AmazonS3ReadOnlyAccess	Amazon S3 更新 AmazonS3FullAccess 和 AmazonS3ReadOnlyAccess 策略以包括 S3 对象 Lambda 的权限。	2021 年 9 月 27 日
Amazon S3 添加 AmazonS3ObjectLambdaExecutionRolePolicy	Amazon S3 添加了一个新的 AWS 托管策略，称为 AmazonS3ObjectLambdaExecutionRolePolicy ，它提供了 Lambda 函数权限，以便与 S3 对象 Lambda 交互并写入 CloudWatch Logs。	2021 年 8 月 18 日
Amazon S3 开始跟踪更改	Amazon S3 为其 AWS 托管策略开启了跟踪更改。	2021 年 8 月 18 日

将服务相关角色用于 Amazon S3 Storage Lens 存储统计管理工具

要使用 Amazon S3 Storage Lens 存储统计管理工具来收集和聚合 AWS Organizations 中所有账户的指标，您必须首先确保 S3 Storage Lens 存储统计管理工具具有由企业中的管理账户启用的可信访问权限。S3 Storage Lens 存储统计管理工具将创建一个服务相关角色，使其能够获取属于您组织的 AWS 账户的列表。在创建或更新 S3 Storage Lens 存储统计管理工具控制面板或配置时，S3 Storage Lens 存储统计管理工具使用此账户列表来收集所有成员账户中 S3 资源的指标。

Amazon S3 Storage Lens 存储统计管理工具使用 AWS Identity and Access Management（IAM）服务相关角色。服务相关角色是一种与 S3 Storage Lens 存储统计管理工具直接关联的独特类型的 IAM 角色。服务相关角色由 S3 Storage Lens 存储统计管理工具预定义，具有服务代表您调用其它 AWS 服务所需的所有权限。

服务相关角色可让您更轻松地设置 S3 Storage Lens 存储统计管理工具，因为您不必手动添加必要的权限。S3 Storage Lens 存储统计管理工具定义了其服务相关角色的权限，除非另外定义，否则只有 S3 Storage Lens 存储统计管理工具可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在先删除相关资源后，才能删除此服务相关角色。这将保护您的 S3 Storage Lens 存储统计管理工具资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的 AWS 服务，并查找 Service-Linked Role（服务相关角色）列为 Yes（是）的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。

Amazon S3 Storage Lens 存储统计管理工具的服务相关角色权限

S3 Storage Lens 存储统计管理工具使用名为 AWSServiceRoleForS3StorageLens 的服务相关角色 – 这支持访问 S3 Storage Lens 存储统计管理工具使用或管理的 AWS 服务和资源。这允许 S3 Storage Lens 存储统计管理工具代表您访问 AWS Organizations 资源。

S3 Storage Lens 存储统计管理工具服务相关角色信任组织存储上的以下服务：

- `storage-lens.s3.amazonaws.com`

角色权限策略允许 S3 Storage Lens 存储统计管理工具完成以下操作：

- `organizations:DescribeOrganization`

  `organizations:ListAccounts`

```
organizations:ListAWSServiceAccessForOrganization

organizations:ListDelegatedAdministrators
```

您必须配置权限以允许 IAM 实体（例如，用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅 IAM 用户指南中的服务相关角色权限。

为 S3 Storage Lens 存储统计管理工具创建服务相关角色

您无需手动创建服务相关角色。当您在登录 AWS Organizations 管理或委托管理员账户时完成以下任务之一后，S3 Storage Lens 存储统计管理工具将为您创建服务相关角色：

- 在 Amazon S3 控制台中为您的组织创建 S3 Storage Lens 存储统计管理工具控制面板配置。
- 使用 REST API、AWS CLI 和 SDK 为您的企业 PUT S3 Storage Lens 存储统计管理工具配置。

> **Note**
>
> S3 Storage Lens 存储统计管理工具最多将为每个组织支持五名委派管理员。

如果删除此服务相关角色，则上述操作将根据需要重新创建该角色。

S3 Storage Lens 存储统计管理工具服务相关角色的示例策略

Example S3 Storage Lens 存储统计管理工具服务相关角色的权限策略

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AwsOrgsAccess",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
```

```
                "*"
            ]
        }
    ]
}
```

为 Amazon S3 Storage Lens 存储统计管理工具编辑服务相关角色

S3 Storage Lens 存储统计管理工具不允许您编辑 AWSServiceRoleForS3StorageLens 服务相关角色。在创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。不过，您可以使用 IAM 编辑角色的说明。有关更多信息，请参阅 IAM 用户指南中的编辑服务相关角色。

删除 Amazon S3 Storage Lens 存储统计管理工具的服务相关角色

如果您不再使用服务相关角色，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除服务相关角色的资源，然后才能手动删除它。

> **Note**
>
> 如果在您试图删除资源时 Amazon S3 Storage Lens 存储统计管理工具服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，则请等待几分钟后重试。

要删除 AWSServiceRoleForS3StorageLens，您必须使用 AWS Organizations 管理或委派管理员账户删除所有 AWS 区域中存在的所有组织级别 S3 Storage Lens 存储统计管理工具配置。

这些资源是组织级 S3 Storage Lens 存储统计管理工具配置。使用 S3 Storage Lens 存储统计管理工具清理资源，然后使用 IAM 控制台、CLI、REST API 或 AWS SDK 删除角色。

在 REST API、AWS CLI 和 SDK 中，可以在组织创建 S3 Storage Lens 存储统计管理工具配置的所有区域中使用 `ListStorageLensConfigurations` 发现 S3 Storage Lens 存储统计管理工具配置。使用操作 `DeleteStorageLensConfiguration` 删除这些配置，以便随后可以删除角色。

> **Note**
>
> 要删除服务相关角色，您必须在存在这些配置的所有区域中删除所有组织级 S3 Storage Lens 配置。

删除 AWSServiceRoleForS3StorageLens SLR 使用的 Amazon S3 Storage Lens 存储统计管理工具资源

1. 要获取您的组织级别配置的列表，您必须在具有 S3 Storage Lens 存储统计管理工具配置的每个区域中使用 `ListStorageLensConfigurations`。此列表也可以从 Amazon S3 控制台获取。
2. 通过调用 `DeleteStorageLensConfiguration` API 调用或通过使用 Amazon S3 控制台从适当的区域端点中删除这些配置。

使用 IAM 手动删除服务相关角色

删除配置后，可以从 IAM 控制台中，或通过调用 IAM API `DeleteServiceLinkedRole`，或使用 AWS CLI 或 AWS SDK 删除 AWSServiceRoleForS3StorageLens SLR。有关更多信息，请参阅 IAM 用户指南中的删除服务相关角色。

S3 Storage Lens 存储统计管理工具服务相关角色的受支持区域

S3 Storage Lens 存储统计管理工具支持在提供服务相关角色的所有 AWS 区域中使用该服务。有关更多信息，请参阅 Amazon S3 区域和端点。

Amazon S3 身份和访问问题排查

可以使用以下信息，协助您诊断和修复在使用 Amazon S3 和 IAM 时可能遇到的常见问题。

主题

- 我收到了拒绝访问错误
- 我无权在 Amazon S3 中执行操作
- 我无权执行 iam:PassRole
- 我希望允许我的 AWS 账户以外的人访问我的 Amazon S3 资源

我收到了拒绝访问错误

确认无论是在存储桶策略还是基于身份的策略中，都没有针对您尝试向其授予权限的请求者使用显式 `Deny` 语句。

有关排查拒绝访问错误的详细信息，请参阅排查 Amazon S3 中的拒绝访问（403 Forbidden）错误。

我无权在 Amazon S3 中执行操作

如果您收到错误提示，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 s3:*GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 s3:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 s3:*GetWidget* 操作访问 *my-example-widget* 资源。

如果您需要帮助，请联系 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam:PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 Amazon S3。

有些 AWS 服务 允许将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 marymajor 的 IAM 用户尝试使用控制台在 Amazon S3 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系 AWS 管理员。您的管理员是提供登录凭证的人。

我希望允许我的 AWS 账户以外的人访问我的 Amazon S3 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon S3 是否支持这些功能，请参阅Amazon S3 如何与 IAM 配合使用。
- 要了解如何为您拥有的 AWS 账户中的资源提供访问权限，请参阅《IAM 用户指南》中的为您拥有的另一个 AWS 账户中的 IAM 用户提供访问权限。

- 要了解如何为第三方 AWS 账户 提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的 AWS 账户 提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用 S3 Access Grants 管理访问权限

要遵守最低权限原则，您可以根据应用程序、角色、组或组织单位来定义对 Amazon S3 数据的精细访问权限。根据访问模式的规模和复杂性，可以使用各种方法实现对 Amazon S3 中的数据的精细访问。

对于 AWS Identity and Access Management（IAM）主体来说，通过定义 IAM 权限策略和 S3 存储桶策略，可以最轻松地管理对 Amazon S3 中的少量和中等数量数据集的访问。只要必要的策略符合 S3 存储桶策略的策略大小限制（20 KB）和 IAM 策略的策略大小限制（5 KB），并且在每个账户允许的 IAM 主体数量内，此策略就将适用。

随着数据集和使用场景不断增多，您可能需要更多的策略空间。为策略语句提供更多空间的一种方法是，使用 S3 接入点作为 S3 存储桶的附加端点，因为每个接入点均可拥有自己的策略。您可以定义非常精细的访问控制模式，因为每个账户在每个 AWS 区域内可以拥有数千个接入点，每个接入点的策略大小最多 20 KB。尽管 S3 接入点增加了可用的策略空间，但它需要通过一种机制让客户端发现正确数据集的正确接入点。

第三种方法是实施 IAM 会话代理模式，在该模式中，您可以实施访问决策逻辑并为每个访问会话动态生成短期 IAM 会话凭证。虽然 IAM 会话代理方法支持任意动态权限模式并能有效扩展，但您必须构建访问模式逻辑。

您可以使用 S3 Access Grants 来管理对 Amazon S3 数据的访问权限，而不是使用上述方法。S3 Access Grants 提供了一个简化模型，用于按前缀、存储桶或对象定义对 Amazon S3 中数据的访问权限。此外，可以使用 S3 Access Grants 向 IAM 主体授予访问权限，并直接向公司目录中的用户或组授予访问权限。

通常，可以通过将用户和组映射到数据集来定义对 Amazon S3 中数据的权限。可以使用 S3 Access Grants 定义 S3 前缀与 Amazon S3 存储桶和对象中的用户与角色的直接访问映射。借助 S3 Access Grants 中的简化访问方案，可以按 S3 前缀向 IAM 主体授予只读、只写或读写访问权限，并直接向公司目录中的用户或组授予只读、只写或读写访问权限。借助这些 S3 Access Grants 功能，应用程序可以代表其当前经过身份验证的用户向 Amazon S3 请求数据。

将 S3 Access Grants 与 AWS IAM Identity Center 的可信身份传播功能集成后，应用程序可以直接代表经过身份验证的公司目录用户向 AWS 服务（包括 S3 Access Grants）发出请求。应用程序不再需要先将用户映射到 IAM 主体。此外，由于最终用户身份将一直传播到 Amazon S3，因此简化了审核哪个用户访问了哪个 S3 对象的过程。您不再需要重构不同的用户和 IAM 会话之间的关系。在将 S3 Access Grants 与 IAM Identity Center 可信身份传播结合使用时，Amazon S3 的每个 AWS CloudTrail 数据事件均包含对代表其访问数据的最终用户的直接引用。

有关 S3 Access Grants 的更多信息，请参阅以下主题。

主题

S3 Access Grants 的概念

S3 访问权限管控工作流程

S3 访问权限管控工作流程为：

1. 创建 S3 访问权限管控实例。请参阅 创建 S3 访问权限管控实例。
2. 在 S3 访问权限管控实例中，在 Amazon S3 数据中注册位置，并将这些位置映射到 AWS Identity and Access Management（IAM）角色。请参阅 注册位置。
3. 为被授权者创建授权，这会向被授权者授予访问 S3 资源的权限。请参阅 创建授权。
4. 被授权者从 S3 访问权限管控请求临时凭证。请参阅 通过 S3 Access Grants 请求访问 Amazon S3 数据。

5. 被授权者使用这些临时凭证来访问 S3 数据。请参阅 通过访问授权来访问 S3 数据。

有关更多信息，请参阅 开始使用 S3 Access Grants。

S3 Access Grants 实例

S3 访问权限管控实例 是各个授权 的逻辑容器。创建 S3 访问权限管控实例时，您必须指定 AWS 区域。您 AWS 账户中的每个 AWS 区域都可具有一个 S3 访问权限管控实例。有关更多信息，请参阅 创建 S3 访问权限管控实例。

如果您想使用 S3 访问权限管控向公司目录中的用户和组身份授予访问权限，还必须将 S3 访问权限管控实例与 AWS IAM Identity Center 实例关联。有关更多信息，请参阅 S3 Access Grants 和公司目录身份。

新创建的 S3 访问权限管控实例为空。您必须在实例中注册一个位置，该位置可以是 S3 默认路径 (s3://)、存储桶或存储桶内的前缀。在您注册至少一个位置后，您可以创建访问权限管控，来授予访问该已注册位置中的数据的权限。

位置

S3 访问权限管控位置 将存储桶或前缀映射到 AWS Identity and Access Management（IAM）角色。S3 访问权限管控代入此 IAM 角色，来向访问该特定位置的被授权者提供临时凭证。您必须先在 S3 访问权限管控实例中注册至少一个位置，然后才能创建访问权限管控。

我们建议您注册默认位置 (s3://) 并将其映射到 IAM 角色。默认 S3 路径 (s3://) 处的位置涵盖了对您账户的 AWS 区域中所有 S3 存储桶的访问权限。创建访问权限管控时，您可以将授予范围缩小到默认位置内的存储桶、前缀或对象。

更复杂的访问管理用例可能要求您注册比默认位置更多的位置。此类用例的一些示例为：

- 假设 *amzn-s3-demo-bucket* 是 S3 访问权限管控实例中的一个注册位置，并且有一个 IAM 角色映射到该位置，但拒绝该 IAM 角色访问存储桶中的特定前缀。在这种情况下，您可以将 IAM 角色无权访问的前缀注册为单独的位置，并将该位置映射到具有必要访问权限的其它 IAM 角色。
- 假设您要创建的授权将访问权限仅限制为虚拟私有云（VPC）端点中的用户。在这种情况下，您可以为存储桶注册一个位置，IAM 角色在该位置限制对 VPC 端点的访问。稍后，当被授权者向 S3 访问权限管控索取凭证时，S3 访问权限管控将代入该位置的 IAM 角色来提供临时凭证。除非调用方位于 VPC 端点内，否则此凭证将拒绝访问特定存储桶。除在授权中指定的常规 READ、WRITE 或 READWRITE 权限外，还会应用此拒绝权限。

如果您的用例要求您在 S3 访问权限管控实例中注册多个位置，则可以注册以下任意位置：

- 默认 S3 位置 (s3://)

- 一个存储桶（例如 *amzn-s3-demo-bucket*）或多个存储桶
- 一个存储桶和一个前缀（例如 *amzn-s3-demo-bucket*/prefix*）或多个前缀

有关您可以在 S3 访问权限管控实例中注册的最大位置数，请参阅 S3 Access Grants 限制。有关注册 S3 访问权限管控位置的更多信息，请参阅注册位置。

在 S3 访问权限管控实例中注册第一个位置后，您的实例中仍没有任何单个访问权限管控。因此，尚未授予对任何 S3 数据的访问权限。现在，您可以创建访问权限管控来授予访问权限。有关创建授权的更多信息，请参阅创建授权。

授权

S3 访问权限管控实例中的单个授权 可让特定身份（IAM 主体或公司目录中的用户或组）在 S3 访问权限管控实例中注册的位置内获得访问权限。

创建授权时，您不必授予对整个已注册位置的访问权限。您可以缩小某个位置内授权的访问权限范围。如果注册位置是默认 S3 路径 (s3://)，则需要将授权的范围缩小到存储桶、存储桶内的前缀或特定对象。如果授权的注册位置是存储桶或前缀，则可以授予对整个存储桶或前缀的访问权限，也可以选择将授权范围缩小到前缀、子前缀或对象。

在授权中，还将授权的访问权限级别设置为 READ、WRITE 或 READWRITE。假设您拥有授权，用于向公司目录组 01234567-89ab-cdef-0123-456789abcdef 授予对存储桶 s3://*amzn-s3-demo-bucket*/projects/items/* 的 READ 访问权限。对于名为 *amzn-s3-demo-bucket* 的存储桶中对象键名称以前缀 projects/items/ 开头的每个对象，该组中的用户都可以具有 READ 访问权限。

有关您可以在 S3 访问权限管控实例中创建的最大授权数量，请参阅 S3 Access Grants 限制。有关创建授权的更多信息，请参阅创建授权。

S3 Access Grants 临时凭证

创建授权后，利用在授权中指定的身份的获授权应用程序可以请求即时访问凭证。为此，应用程序调用 GetDataAccess S3 API 操作。被授权者可以使用此 API 操作来请求访问您与他们共享的 S3 数据。

S3 Access Grants 实例根据其拥有的授权来评估 GetDataAccess 请求。如果对于请求者存在匹配的授权，则 S3 访问权限管控将代入与匹配授权的已注册位置关联的 IAM 角色。S3 访问权限管控将临时凭证的权限范围限定为仅访问由授权的范围指定的 S3 存储桶、前缀或对象。

虽然临时访问凭证的到期时间默认为 1 小时，但可以将到期时间设置为介于 15 分钟和 12 小时之间的任意值。请参阅 AssumeRole API 参考中的最长持续时间会话。

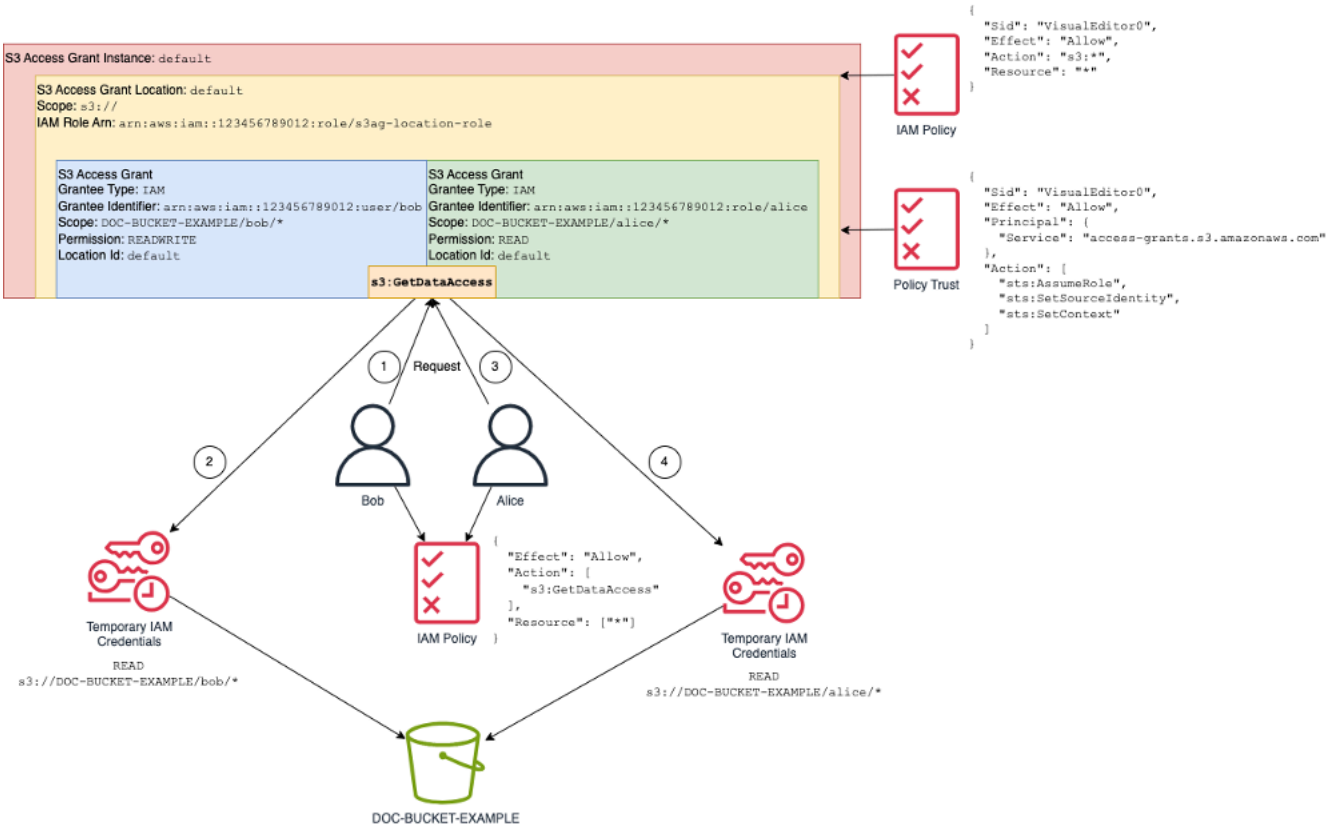
工作方式

在下图中，范围为 s3:// 的默认 Amazon S3 位置已注册到 IAM 角色 s3ag-location-role。当通过 S3 Access Grants 获取凭证时，此 IAM 角色有权在账户内执行 Amazon S3 操作。

在此位置，为两个 IAM 用户创建两个单独的访问授权。向 IAM 用户 Bob 授予对 DOC-BUCKET-EXAMPLE 存储桶中的 bob/ 前缀的 READ 和 WRITE 访问权限。仅向另一个 IAM 角色 Alice 授予对 DOC-BUCKET-EXAMPLE 桶中 alice/ 前缀的 READ 访问权限。为 Bob 定义了用于访问 DOC-BUCKET-EXAMPLE 存储桶中的前缀 bob/ 的授权（蓝色）。为 Alice 定义了用于访问 DOC-BUCKET-EXAMPLE 存储桶中的前缀 alice/ 的授权（绿色）。

当 Bob 需要对数据进行 READ 访问时，与其授权所在位置关联的 IAM 角色会调用 S3 Access Grants GetDataAccess API 操作。如果 Bob 尝试对任何以 s3://DOC-BUCKET-EXAMPLE/bob/* 开头的 S3 前缀或对象进行 READ 访问，则 GetDataAccess 请求将返回一组具有对 s3://DOC-BUCKET-EXAMPLE/bob/* 的访问权限的临时 IAM 会话凭证。同样，Bob 可以对任何以 s3://DOC-BUCKET-EXAMPLE/bob/* 开头的 S3 前缀或对象进行 WRITE 访问，因为该授权也允许这样做。

同样，Alice 可以对任何以 s3://DOC-BUCKET-EXAMPLE/alice/ 开头的项进行 READ 访问。但是，如果她尝试对 s3:// 中的任何存储桶、前缀或对象进行任何 WRITE 访问，她将收到“拒绝访问（403 禁止）”错误，因为没有允许她对任何数据进行 WRITE 访问的授权。此外，如果 Alice 请求对 s3://DOC-BUCKET-EXAMPLE/alice/ 外部的数据进行任何级别的访问（READ 或 WRITE），她将再次收到“拒绝访问”错误。



此模式可以扩展到大量的用户和存储桶，并简化对这些权限的管理。您可以添加和删除单个离散授权，而不是在每次需要添加或删除单个用户前缀访问关系时编辑可能很大的 S3 存储桶策略。

S3 Access Grants 和公司目录身份

您可以使用 Amazon S3 Access Grants 向 AWS Identity and Access Management（IAM）主体（用户或角色）授予访问权限，无论是在同一个还是其他 AWS 账户中。但在许多情况下，访问数据的实体是公司目录中的最终用户。可以使用 S3 Access Grants 直接向公司用户和组授予访问权限，而不是向 IAM 主体授予访问权限。借助 S3 Access Grants，您不再需要将公司身份映射到中间 IAM 主体，即可通过公司应用程序访问您的 S3 数据。

这项新功能（支持使用最终用户身份访问数据）是通过将 S3 Access Grants 实例与 AWS IAM Identity Center 实例关联来提供的。IAM Identity Center 支持基于标准的身份提供商，并且是 AWS 中支持最终用户身份的任何服务或功能（包括 S3 Access Grants）的中心。IAM Identity Center 通过其可信身份传播功能来支持验证公司身份。有关更多信息，请参阅跨应用程序的可信身份传播。

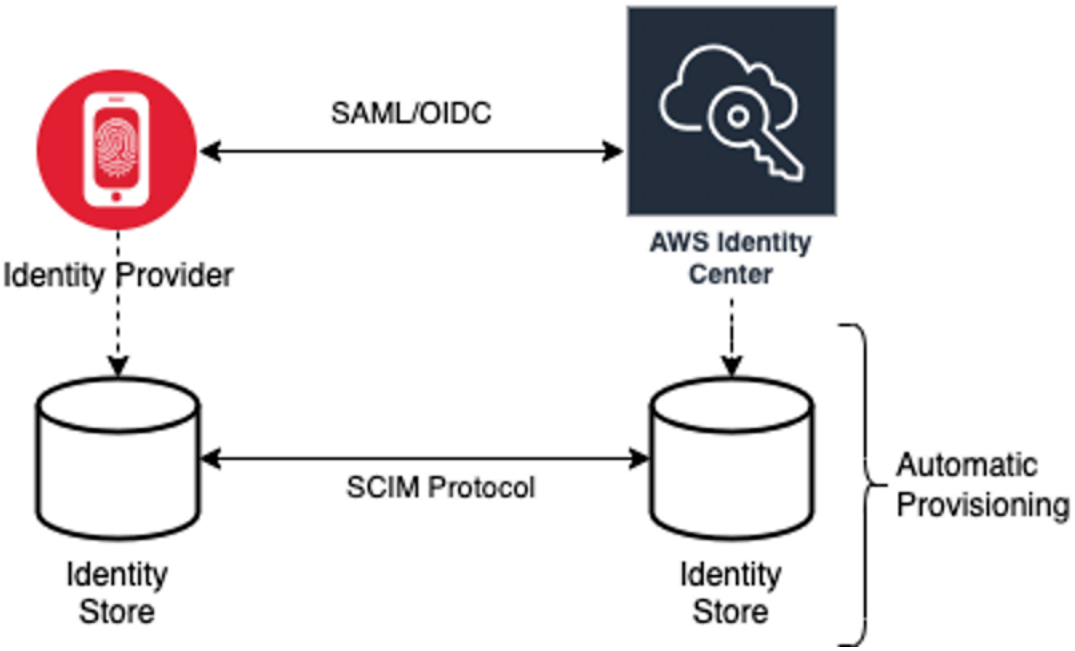
要开始使用 S3 Access Grants 中的人力身份支持，您首先必须从 IAM Identity Center 开始，在公司身份提供商和 IAM Identity Center 之间配置身份预置。IAM Identity Center 支持公司身份提供商，例如 Okta、Microsoft Entra ID（以前称为 Azure Active Directory），或任何其他支持跨域身份管理系统（SCIM）协议的外部身份提供商（IdP）。在将 IAM Identity Center 连接到 IdP 并启用

自动预置时，IdP 中的用户和组将同步到 IAM Identity Center 中的身份存储。此步骤完成后，IAM Identity Center 将拥有自己的用户和组视图，这样您便能使用其他 AWS 服务 和功能（例如 S3 Access Grants）来引用它们。有关配置 IAM Identity Center 自动预置的更多信息，请参阅《AWS IAM Identity Center 用户指南》中的自动预置。

IAM Identity Center 与 AWS Organizations 集成，这使您能够集中管理多个 AWS 账户的权限，而无需手动配置每个账户。在典型企业中，您的身份管理员会为整个企业配置一个 IAM Identity Center 实例，以作为单个身份同步点。此 IAM Identity Center 实例通常在企业的专用 AWS 账户中运行。在此通用配置中，您可以从企业内的任何 AWS 账户中引用 S3 Access Grants 中的用户和组身份。

但是，如果 AWS Organizations 管理员尚未配置中央 IAM Identity Center 实例，您可以在与 S3 Access Grants 实例相同的账户中创建一个本地实例。此类配置在概念验证或本地开发使用场景中更常见。在所有情况下，IAM Identity Center 实例都必须在与其关联到的 S3 Access Grants 实例相同的 AWS 区域内。

在下面的 IAM Identity Center 配置与外部 IdP 图中，已使用 SCIM 配置 IdP，以将身份存储从 IdP 同步到 IAM Identity Center 中的身份存储。



要将您的公司目录身份与 S3 Access Grants 结合使用，请执行以下操作：

- 在 IAM Identity Center 中设置自动预置，将来自 IdP 的用户和组信息同步到 IAM Identity Center。
- 将您在 IAM Identity Center 中的外部身份源配置为可信令牌颁发机构。有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的跨应用程序的可信身份传播。

- 将 S3 Access Grants 实例与 IAM Identity Center 实例关联。您可以在创建 S3 Access Grants 实例时执行此操作。如果您已创建 S3 Access Grants 实例，请参阅 关联或取消关联 IAM Identity Center 实例。

目录身份访问 S3 数据的方式

假设您的公司目录用户需要通过公司应用程序（例如文档查看器应用程序）访问您的 S3 数据，该应用程序与您的外部 IdP（例如 Okta）集成以对用户进行身份验证。这些应用程序中的用户身份验证通常将通过用户 Web 浏览器中的重定向来完成。由于目录中的用户不是 IAM 主体，因此您的应用程序需要 IAM 凭证，该凭证可用于代表用户调用 S3 Access Grants `GetDataAccess` API 操作来获取 S3 数据的访问凭证。与自行获得凭证的 IAM 用户和角色不同，您的应用程序需要通过某种方式来表示未映射到 IAM 角色的目录用户，以便该用户能够通过 S3 Access Grants 获得数据访问权限。

从经过身份验证的目录用户到能够代表目录用户向 S3 Access Grants 发出请求的 IAM 调用方是一个过渡，它由应用程序通过 IAM Identity Center 的可信令牌颁发机构功能完成。在对目录用户进行身份验证后，应用程序将获得来自 IdP（例如 Okta）的身份令牌，该令牌代表目录用户，具体取决于 Okta。IAM Identity Center 中的可信令牌颁发机构配置使应用程序能够用此 Okta 令牌（Okta 租户配置为“可信颁发机构”）交换来自 IAM Identity Center 的其他身份令牌，从而安全地表示 AWS 服务中的目录用户。之后，数据应用程序将代入一个 IAM 角色，提供来自 IAM Identity Center 的目录用户令牌作为附加上下文。应用程序可使用生成的 IAM 会话来调用 S3 Access Grants。令牌既表示应用程序的身份（IAM 主体本身），也表示目录用户的身份。

此过渡的主要步骤是令牌交换。应用程序通过在 IAM Identity Center 中调用 `CreateTokenWithIAM` API 操作来执行此令牌交换。当然，这也是 AWS API 调用，需要一个 IAM 主体进行签名。发出此请求的 IAM 主体通常是与应用程序关联的 IAM 角色。例如，如果应用程序在 Amazon EC2 上运行，则 `CreateTokenWithIAM` 请求通常由与运行该应用程序的 EC2 实例关联的 IAM 角色执行。成功调用 `CreateTokenWithIAM` 会生成一个新的身份令牌，该令牌将在 AWS 服务中被识别。

在应用程序可以代表目录用户调用 `GetDataAccess` 之前，下一步是让应用程序获取包含目录用户身份的 IAM 会话。应用程序通过 AWS Security Token Service（AWS STS）`AssumeRole` 请求执行此操作，该请求还包括目录用户的 IAM Identity Center 令牌作为附加身份上下文。此附加上下文可让 IAM Identity Center 将目录用户的身份传播到下一步。应用程序代入的 IAM 角色需要 IAM 权限才能调用 `GetDataAccess` 操作。

在代入身份持有者 IAM 角色并将目录用户的 IAM Identity Center 令牌作为附加上下文后，应用程序现在便能代表经过身份验证的目录用户向 `GetDataAccess` 发出签名请求。

令牌传播基于以下步骤：

创建 IAM Identity Center 应用程序

首先，在 IAM Identity Center 中创建一个新应用程序。此应用程序将使用一个模板，以便 IAM Identity Center 能够识别您可使用的应用程序设置类型。用于创建应用程序的命令要求您提供 IAM Identity Center 实例 Amazon 资源名称（ARN）、应用程序名称和应用程序提供商 ARN。应用程序提供商是应用程序将用于调用 IAM Identity Center 的 SAML 或 OAuth 应用程序提供商。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息：

```
aws sso-admin create-application \
 --instance-arn "arn:aws:sso:::instance/ssoins-ssoins-1234567890abcdef" \
 --application-provider-arn "arn:aws:sso::aws:applicationProvider/custom" \
 --name MyDataApplication
```

响应：

```
{
   "ApplicationArn": "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
}
```

创建可信令牌颁发机构

现在，您已拥有 IAM Identity Center 应用程序，下一步是配置一个可信令牌颁发机构，它用于将您的 IdP 中的 `IdToken` 值与 IAM Identity Center 令牌交换。在此步骤中，您需要提供以下各项：

- 身份提供商颁发机构 URL
- 可信令牌颁发机构名称
- 声明属性路径
- 身份存储属性路径
- JSON Web 密钥集（JWKS）检索选项

声明属性路径是将用于映射到身份存储属性的身份提供商属性。通常，虽然声明属性路径是用户的电子邮件地址，但您可以使用其他属性来执行映射。

创建一个名为 `oidc-configuration.json` 的文件，其中包含以下信息。要使用此文件，请将 *user input placeholders* 替换为您自己的信息。

```
{
```

```
  "OidcJwtConfiguration":
     {
      "IssuerUrl": "https://login.microsoftonline.com/a1b2c3d4-abcd-1234-b7d5-
b154440ac123/v2.0",
      "ClaimAttributePath": "preferred_username",
      "IdentityStoreAttributePath": "userName",
      "JwksRetrievalOption": "OPEN_ID_DISCOVERY"
     }
}
```

要创建可信的令牌颁发机构，请运行以下命令。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws sso-admin create-trusted-token-issuer \
  --instance-arn "arn:aws:sso:::instance/ssoins-1234567890abcdef" \
  --name MyEntraIDTrustedIssuer \
  --trusted-token-issuer-type OIDC_JWT \
  --trusted-token-issuer-configuration file://./oidc-configuration.json
```

响应

```
{
  "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234"
}
```

将 IAM Identity Center 应用程序与可信令牌颁发机构连接

可信令牌颁发机构需要更多的配置设置才能工作。设置可信令牌颁发机构将信任的受众。受众是由密钥标识的 IdToken 内部的值，可以在身份提供商设置中找到它。例如：

```
1234973b-abcd-1234-abcd-345c5a9c1234
```

创建一个名为 grant.json 的文件，其中包含以下内容。要使用此文件，请更改受众以匹配您的身份提供商设置，并提供上一条命令返回的可信令牌颁发机构 ARN。

```
{
   "JwtBearer":
     {
       "AuthorizedTokenIssuers":
```

```
        [
          {
            "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234",
              "AuthorizedAudiences":
                [
                  "1234973b-abcd-1234-abcd-345c5a9c1234"
                ]
           }
        ]
    }
 }
```

运行以下示例命令。要使用此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws sso-admin put-application-grant \
  --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
  --grant-type "urn:ietf:params:oauth:grant-type:jwt-bearer" \
  --grant file://./grant.json \
```

此命令使用配置设置来设定可信令牌颁发机构以信任 `grant.json` 文件中的受众，并将该受众与第一步中创建的应用程序关联起来以便交换类型 `jwt-bearer` 的令牌。字符串 `urn:ietf:params:oauth:grant-type:jwt-bearer` 不是任意字符串。它是 OAuth JSON Web 令牌（JWT）断言配置文件中的注册命名空间。您可以在 RFC 7523 中找到有关此命名空间的更多信息。

接下来，使用以下命令来设置可信令牌颁发机构在交换身份提供商提供的 `IdToken` 值时将包含的范围。对于 S3 Access Grants，`--scope` 参数的值为 `s3:access_grants:read_write`。

```
aws sso-admin put-application-access-scope \
  --application-arn "arn:aws:sso::111122223333:application/ssoins-
ssoins-111122223333abcdef/apl-abcd1234a1b2c3d" \
  --scope "s3:access_grants:read_write"
```

最后一步是将资源策略附加到 IAM Identity Center 应用程序。此策略将允许您的应用程序 IAM 角色向 API 操作 `sso-oauth:CreateTokenWithIAM` 发出请求，并接受来自 IAM Identity Center 的 `IdToken` 值。

创建一个名为 `authentication-method.json` 的文件，其中包含以下内容。请将 *123456789012* 替换为您的账户 ID。

```
{
   "Iam":
       {
         "ActorPolicy":
             {
                "Version": "2012-10-17",
                    "Statement":
                    [
                        {
                           "Effect": "Allow",
                            "Principal":
                            {
                              "AWS": "arn:aws:iam::123456789012:role/webapp"
                            },
                           "Action": "sso-oauth:CreateTokenWithIAM",
                            "Resource": "*"
                        }
                    ]
                }
            }
        }
```

要将策略附加到 IAM Identity Center 应用程序，请运行以下命令：

```
aws sso-admin put-application-authentication-method \
   --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
   --authentication-method-type IAM \
   --authentication-method file://./authentication-method.json
```

这将完成通过 Web 应用程序对目录用户使用 S3 Access Grants 的配置设置。您可以直接在应用程序中测试此设置，也可以使用以下命令从 IAM Identity Center 应用程序策略中允许的 IAM 角色调用 `CreateTokenWithIAM` API 操作：

```
aws sso-oidc create-token-with-iam \
   --client-id "arn:aws:sso::123456789012:application/ssoins-ssoins-1234567890abcdef/
apl-abcd1234a1b2c3d"  \
   --grant-type urn:ietf:params:oauth:grant-type:jwt-bearer \
   --assertion IdToken
```

响应将与以下内容类似：

```
{
    "accessToken": "<suppressed long string to reduce space>",
    "tokenType": "Bearer",
    "expiresIn": 3600,
    "refreshToken": "<suppressed long string to reduce space>",
    "idToken": "<suppressed long string to reduce space>",
    "issuedTokenType": "urn:ietf:params:oauth:token-type:refresh_token",
    "scope": [
      "sts:identity_context",
      "s3:access_grants:read_write",
      "openid",
      "aws"
    ]
}
```

如果您对使用 base64 编码的 IdToken 值进行解码，则会看到采用 JSON 格式的键值对。密钥 sts:identity_context 包含应用程序需要在 sts:AssumeRole 请求中发送的值，以包含目录用户的身份信息。以下是解码的 IdToken 的示例：

```
{
    "aws:identity_store_id": "d-996773e796",
    "sts:identity_context": "AQoJb3JpZ2luX2VjEOTtl;<SUPRESSED>",
    "sub": "83d43802-00b1-7054-db02-f1d683aacba5",
    "aws:instance_account": "123456789012",
    "iss": "https://identitycenter.amazonaws.com/ssoins-1234567890abcdef",
    "sts:audit_context": "AQoJb3JpZ2luX2VjEOT<SUPRESSED>==",
    "aws:identity_store_arn": "arn:aws:identitystore::232642235904:identitystore/
d-996773e796",
    "aud": "abcd12344U0gi7n4Yyp0-WV1LWNlbnRyYWwtMQ",
    "aws:instance_arn": "arn:aws:sso:::instance/ssoins-6987d7fb04cf7a51",
    "aws:credential_id": "EXAMPLEHI5glPh40y9TpApJn8...",
    "act": {
       "sub": "arn:aws:sso::232642235904:trustedTokenIssuer/
ssoins-6987d7fb04cf7a51/43b4a822-1020-7053-3631-cb2d3e28d10e"
    },
    "auth_time": "2023-11-01T20:24:28Z",
    "exp": 1698873868,
    "iat": 1698870268
}
```

您可以从 sts:identity_context 中获取值并在 sts:AssumeRole 调用中传递该信息。以下是语法的 CLI 示例。要代入的角色是有权调用 s3:GetDataAccess 的临时角色。

```
aws sts assume-role \
   --role-arn "arn:aws:iam::123456789012:role/temp-role" \
   --role-session-name "TempDirectoryUserRole" \
   --provided-contexts ProviderArn="arn:aws:iam::aws:contextProvider/
IdentityCenter",ContextAssertion="value from sts:identity_context"
```

现在，您可以使用从该调用中接收的凭证来调用 `s3:GetDataAccess` API 操作，并接收可用于访问您的 S3 资源的最终凭证。

开始使用 S3 Access Grants

Amazon S3 Access Grants 是 Amazon S3 的一项功能，为 S3 数据提供了可扩展的访问控制解决方案。S3 Access Grants 是 S3 凭证供应商，这意味着您将向 S3 Access Grants 注册您的授权列表和级别。此后，当用户或客户端需要访问您的 S3 数据时，他们首先需向 S3 Access Grants 索要凭证。如果有相应的授权来授予访问权限，则 S3 Access Grants 会分发最低权限的临时访问凭证。之后，用户或客户端可以使用 S3 Access Grants 分发的凭证来访问 S3 数据。考虑到这一点，如果您的 S3 数据要求指示进行复杂或大规模的权限配置，则可以使用 S3 Access Grants 来扩展用户、组、角色和应用程序的 S3 数据权限。

对于大多数使用场景，您可以通过使用带存储桶策略的 AWS Identity and Access Management（IAM）或基于 IAM 身份的策略来管理 S3 数据的访问控制。

不过，如果您有复杂的 S3 访问控制要求（例如以下要求），则可通过使用 S3 Access Grants 获得大量好处：

- 您达到了存储桶策略大小限制 20 KB。
- 您可以向人类身份（例如 Microsoft Entra ID（以前称作 Azure Active Directory）、Okta 或 Ping 用户和组）授予对 S3 数据的访问权限以进行分析和大数据处理。
- 您必须提供跨账户访问，而不是频繁更新 IAM 策略。
- 您的数据是非结构化的对象级数据，而不是采用行和列格式的结构化数据。

S3 Access Grants 工作流如下所示：

步骤	描述
1	创建 S3 访问权限管控实例

步骤	描述
	要开始使用，请启动一个 S3 Access Grants 实例，其中包含您的个人访问授权。
2	注册位置 紧接着，注册一个 S3 数据位置（例如默认位置 `s3://`），然后指定 S3 Access Grants 在提供对 S3 数据位置的访问权限时所代入的默认 IAM 角色。您还可以将自定义位置添加到特定的存储桶或前缀，并将其映射到自定义 IAM 角色。
3	创建授权 创建单个权限授权。在这些权限授权中指定注册的 S3 位置、该位置内的数据访问范围、被授权者的身份及其访问级别（`READ`、`WRITE` 或 `READWRITE` ）。
4	请求对 S3 数据的访问权限 当用户、应用程序和 AWS 服务 需要访问 S3 数据时，他们首先会发出访问请求。S3 Access Grants 决定是否应授权请求。如果有相应的授权来授予访问权限，则 S3 Access Grants 会使用与该授权关联的注册位置的 IAM 角色，以再次将临时凭证分发给请求者。
5	访问 S3 数据 应用程序使用 S3 Access Grants 分发的临时凭证来访问 S3 数据。

创建 S3 访问权限管控实例

要开始使用 Amazon S3 Access Grants，您需要先创建一个 S3 Access Grants 实例。您的每个账户在每个 AWS 区域 中只能创建一个 S3 Access Grants 实例。S3 Access Grants 实例充当您的 S3 Access Grants 资源的容器，其中包括注册的位置和授权。

利用 S3 Access Grants，您可以为 AWS Identity and Access Management（IAM）用户和角色创建对 S3 数据的权限授权。如果您已向 AWS IAM Identity Center 添加公司身份目录，则可以将公司目录的

此 IAM Identity Center 实例与您的 S3 Access Grants 实例相关联。完成此操作后，您可以为公司用户和组创建访问授权。如果您尚未将公司目录添加到 IAM Identity Center，则可以稍后将您的 S3 Access Grants 实例与 IAM Identity Center 实例相关联。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 创建 S3 Access Grants 实例。

使用 S3 控制台

在使用 S3 Access Grants 授予对 S3 数据的访问权限之前，必须先在 S3 数据所在的同一 AWS 区域中创建 S3 Access Grants 实例。

先决条件

如果您想使用公司目录中的身份授予对 S3 数据的访问权限，请向 AWS IAM Identity Center 添加公司身份目录。如果您尚未准备好执行此操作，可以稍后将您的 S3 Access Grants 实例与 IAM Identity Center 实例关联。

创建 S3 访问权限管控实例

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要切换到的区域。
3. 在左侧导航窗格中，选择 Access Grants。
4. 在 S3 Access Grants 页面上，选择创建 S3 Access Grants 实例。

    a. 在设置 Access Grants 实例向导的步骤 1 中，确认您要在当前的 AWS 区域中创建实例。确保这是您的 S3 数据所在的 AWS 区域。您的每个账户在每个 AWS 区域 中可创建一个 S3 Access Grants 实例。

    b. （可选）如果您已向 AWS IAM Identity Center 添加公司身份目录，则可以将公司目录的此 IAM Identity Center 实例与您的 S3 Access Grants 实例相关联。

    为此，请选择在**##**中添加 IAM Identity Center 实例。然后输入 IAM Identity Center 实例 Amazon 资源名称（ARN）。

    如果您尚未将公司目录添加到 IAM Identity Center，则可以稍后将您的 S3 Access Grants 实例与 IAM Identity Center 实例相关联。

    c. 要创建 S3 Access Grants 实例，请选择下一步。要注册位置，请参阅步骤 2 – 注册位置。
5. 如果已禁用下一步或创建 S3 Access Grants 实例，则：

无法创建实例

- 您可能已在同一个 AWS 区域中拥有一个 S3 Access Grants 实例。在左侧导航窗格中，选择 Access Grants。在 S3 Access Grants 页面上，向下滚动到您的账户中的 S3 Access Grants 实例以确定是否已存在实例。
- 您可能没有创建 S3 Access Grants 实例所需的 `s3:CreateAccessGrantsInstance` 权限。请联系您的账户管理员。有关将 IAM Identity Center 实例与 S3 Access Grants 实例关联时所需的其他权限，请参阅 CreateAccessGrantsInstance。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *`user input placeholders`* 替换为您自己的信息。

Example 创建 S3 访问权限管控实例

```
aws s3control create-access-grants-instance \
--account-id 111122223333 \
--region us-east-2
```

响应：

```
{
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

使用 REST API

您可以使用 Amazon S3 REST API 创建 S3 Access Grants 实例。有关用于管理 S3 Access Grants 实例的 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- AssociateAccessGrantsIdentityCenter
- CreateAccessGrantsInstance
- DeleteAccessGrantsInstance

- DissociateAccessGrantsIdentityCenter
- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- GetAccessGrantsInstanceResourcePolicy
- ListAccessGrantsInstances
- PutAccessGrantsInstanceResourcePolicy

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 创建 S3 Access Grants 实例。

Java

此示例创建的 S3 Access Grants 实例可用作您的个人访问授权的容器。您的每个账户在每个 AWS 区域 中可具有一个 S3 Access Grants 实例。响应包括实例 ID `default` 和为您的 S3 Access Grants 实例生成的 Amazon 资源名称（ARN）。

Example 创建 S3 Access Grants 实例请求

```
public void createAccessGrantsInstance() {
CreateAccessGrantsInstanceRequest createRequest =
 CreateAccessGrantsInstanceRequest.builder().accountId("111122223333").build();
CreateAccessGrantsInstanceResponse createResponse =
 s3Control.createAccessGrantsInstance(createRequest);LOGGER.info("CreateAccessGrantsInstance
 " + createResponse);
}
```

响应：

```
CreateAccessGrantsInstanceResponse(
CreatedAt=2023-06-07T01:46:20.507Z,
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default)
```

主题

- 查看 S3 Access Grants 实例的详细信息
- 关联或取消关联 IAM Identity Center 实例

- 删除 S3 Access Grants 实例

查看 S3 Access Grants 实例的详细信息

您可以查看特定的 AWS 区域中的 Amazon S3 Access Grants 实例的详细信息。您还可以列出您的 S3 Access Grants 实例，包括已通过 AWS Resource Access Manager（AWS RAM）与您共享的实例。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 来查看 S3 Access Grants 实例的详细信息或列出这些实例。

使用 S3 控制台

查看 S3 Access Grants 实例

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。
4. S3 Access Grants 页面列出了您的 S3 Access Grants 实例以及已与您的账户共享的所有跨账户实例。要查看实例的详细信息，请选择查看详细信息。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 获取 S3 Access Grants 实例的详细信息

```
aws s3control get-access-grants-instance \
 --account-id 111122223333 \
 --region us-east-2
```

响应：

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
```

```
}
```

Example – 列出账户的所有 S3 Access Grants 实例

此操作将列出账户的 S3 Access Grants 实例。您在每个 AWS 区域中只能具有一个 S3 Access Grants 实例。此操作还会列出您的账户有权访问的其他跨账户 S3 Access Grants 实例。

```
aws s3control list-access-grants-instances \
 --account-id 111122223333 \
 --region us-east-2
```

响应：

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

使用 REST API

有关用于管理 S3 Access Grants 实例的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- ListAccessGrantsInstances

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 获取 S3 Access Grants 实例的详细信息。

要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

Java

Example – 获取 S3 Access Grants 实例

```
public void getAccessGrantsInstance() {
```

```
GetAccessGrantsInstanceRequest getRequest = GetAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
GetAccessGrantsInstanceResponse getResponse =
 s3Control.getAccessGrantsInstance(getRequest);
LOGGER.info("GetAccessGrantsInstanceResponse: " + getResponse);
}
```

响应：

```
GetAccessGrantsInstanceResponse(
AccessGrantsInstanceArn=arn:aws:s3:us-east-2: 111122223333:access-grants/default,
CreatedAt=2023-06-07T01:46:20.507Z)
```

Example – 列出账户的所有 S3 Access Grants 实例

此操作将列出账户的 S3 Access Grants 实例。您在每个区域只能有一个 S3 Access Grants 实例。此操作还列出了您的账户有权访问的其他跨账户 S3 Access Grants 实例。

```
public void listAccessGrantsInstances() {
ListAccessGrantsInstancesRequest listRequest =
 ListAccessGrantsInstancesRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsInstancesResponse listResponse =
 s3Control.listAccessGrantsInstances(listRequest);
LOGGER.info("ListAccessGrantsInstancesResponse: " + listResponse);
}
```

响应：

```
ListAccessGrantsInstancesResponse(
AccessGrantsInstancesList=[
ListAccessGrantsInstanceEntry(
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default,
CreatedAt=2023-06-07T04:28:11.728Z
)
]
)
```

关联或取消关联 IAM Identity Center 实例

在 Amazon S3 Access Grants 中，您可以将公司身份目录的 AWS IAM Identity Center 实例与 S3 Access Grants 实例相关联。完成此操作后，除了 AWS Identity and Access Management（IAM）用户和角色之外，您还可以为公司目录用户和组创建访问授权。

如果您不再想为公司目录用户和组创建访问授权，则可以取消您的 IAM Identity Center 实例与 S3 Access Grants 实例的关联。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 来关联或取消关联 IAM Identity Center 实例。

使用 S3 控制台

在将 IAM Identity Center 实例与 S3 Access Grants 实例关联之前，您必须将公司身份目录添加到 IAM Identity Center。有关更多信息，请参阅 the section called “S3 Access Grants 和公司目录身份”。

将 IAM Identity Center 实例与 S3 Access Grants 实例关联

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。
4. 对于实例，选择查看详细信息。
5. 在详细信息页面上的 IAM Identity Center 部分中，选择添加 IAM Identity Center 实例或取消注册已关联的 IAM Identity Center 实例。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 将 IAM Identity Center 实例与 S3 Access Grants 实例关联

```
aws s3control associate-access-grants-identity-center \
 --account-id 111122223333 \
 --identity-center-arn arn:aws:sso:::instance/ssoins-1234a567bb89012c \
 --profile access-grants-profile \
 --region eu-central-1
```

```
// No response body
```

Example – 取消 IAM Identity Center 实例与 S3 Access Grants 实例的关联

```
aws s3control dissociate-access-grants-identity-center \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

使用 REST API

有关用于管理 IAM Identity Center 实例与 S3 Access Grants 实例之间的关联的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- AssociateAccessGrantsIdentityCenter
- DissociateAccessGrantsIdentityCenter

删除 S3 Access Grants 实例

您可以从账户中的 AWS 区域中删除 Amazon S3 Access Grants 实例。但是，您必须先执行以下操作，之后才能删除 S3 Access Grants 实例：

- 删除 S3 Access Grants 实例中的所有资源，包括所有授权和位置。有关更多信息，请参阅删除授权和删除位置。
- 如果您已将 AWS IAM Identity Center 实例与 S3 Access Grants 实例关联，则必须取消关联 IAM Identity Center 实例。有关更多信息，请参阅关联或取消关联 IAM Identity Center 实例。

> **Important**
>
> 如果您删除 S3 Access Grants 实例，则删除是永久性的，无法撤消。通过此 S3 Access Grants 实例中的授权获得访问权限的所有被授权者都将失去对您的 S3 数据的访问权限。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 删除 S3 Access Grants 实例。

使用 S3 控制台

删除 S3 Access Grants 实例

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。
4. 对于实例，选择查看详细信息。
5. 在实例详细信息页面上，选择右上角的删除实例。
6. 在出现的对话框中，选择删除。此操作无法撤消。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

> **Note**
>
> 您必须先删除在 S3 Access Grants 实例中创建的所有授权和位置，之后才能删除 S3 Access Grants 实例。如果您已将 IAM Identity Center 实例与 S3 Access Grants 实例关联，则必须先取消此关联。

Example – 删除 S3 Access Grants 实例

```
aws s3control delete-access-grants-instance \
--account-id 111122223333 \
--profile access-grants-profile \
--region us-east-2 \
--endpoint-url https://s3-control.us-east-2.amazonaws.com \

 // No response body
```

使用 REST API

有关用于删除 S3 Access Grants 实例的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的 DeleteAccessGrantsInstance。

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 删除 S3 Access Grants 实例。

要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

Java

> **Note**
>
> 您必须先删除在 S3 Access Grants 实例中创建的所有授权和位置，之后才能删除 S3 Access Grants 实例。如果您已将 IAM Identity Center 实例与 S3 Access Grants 实例关联，则必须先取消此关联。

Example – 删除 S3 Access Grants 实例

```
public void deleteAccessGrantsInstance() {
DeleteAccessGrantsInstanceRequest deleteRequest =
 DeleteAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
DeleteAccessGrantsInstanceResponse deleteResponse =
 s3Control.deleteAccessGrantsInstance(deleteRequest);
LOGGER.info("DeleteAccessGrantsInstanceResponse: " + deleteResponse);
}
```

注册位置

在您账户的 AWS 区域中创建 Amazon S3 访问权限管控实例后，在该实例中注册 S3 位置。S3 访问权限管控位置将默认 S3 位置 (`s3://`)、存储桶或前缀映射到 AWS Identity and Access Management（IAM）角色。S3 访问权限管控代入此 IAM 角色，来向正在访问该特定位置的被授权者提供临时凭证。您必须先在 S3 访问权限管控实例中注册至少一个位置，然后才能创建访问权限管控。

建议的用例

我们建议您注册默认位置 (`s3://`) 并将其映射到 IAM 角色。默认 S3 路径 (`s3://`) 中的位置涵盖了对您账户的该 AWS 区域中所有 S3 存储桶的访问权限。创建访问权限管控时，您可以将授予范围缩小到默认位置内的存储桶、前缀或对象。

复杂的访问管理用例

更复杂的访问管理用例可能要求您注册比默认位置更多的位置。此类用例的一些示例为：

- 假设 *amzn-s3-demo-bucket* 是 S3 访问权限管控实例中的一个注册位置，并且有一个 IAM 角色映射到该位置，但拒绝该 IAM 角色访问存储桶中的特定前缀。在这种情况下，您可以将 IAM 角色无权访问的前缀注册为单独的位置，并将该位置映射到具有必要访问权限的其它 IAM 角色。
- 假设您要创建的授权将访问权限仅限制为虚拟私有云（VPC）端点中的用户。在这种情况下，您可以为存储桶注册一个位置，IAM 角色在该位置限制对 VPC 端点的访问。稍后，当被授权者向 S3 访问权限管控索取凭证时，S3 访问权限管控将代入该位置的 IAM 角色来提供临时凭证。除非调用方位于 VPC 端点内，否则此凭证将拒绝访问特定存储桶。除在授权中指定的常规 READ、WRITE 或 READWRITE 权限外，还会应用此拒绝权限。

在注册位置时，还必须指定 S3 访问权限管控代入的 IAM 角色，来提供临时凭证和确定特定授权的权限范围。

如果您的用例要求您在 S3 访问权限管控实例中注册多个位置，则可以注册以下任意位置：

S3 URI	IAM 角色	描述
`s3://`	*Default-IAM-role*	默认位置 `s3://` 包括 AWS 区域中的所有存储桶。
`s3://`*amzn-s3-demo-bucket1* `/`	*IAM-role-For-bucket*	此位置包括指定存储桶中的所有对象。
`s3://`*amzn-s3-demo-bucket1* `/`*prefix-name*	*IAM-role-For-prefix*	此位置包括存储桶中对象键名称以此前缀开头的所有对象。

在注册特定存储桶或前缀之前，请确保执行以下操作：

- 创建一个或多个存储桶，其中包含要授予其访问权限的数据。这些存储桶必须位于您的 S3 Access Grants 实例所在的 AWS 区域中。有关更多信息，请参阅创建存储桶。

  添加前缀是一个可选步骤。前缀是位于对象键名称开头的字符串。可以使用它们来组织存储桶中的对象以及进行访问管理。要向存储桶添加前缀，请参阅创建对象密钥名称。
- 创建一个 IAM 角色，该角色有权访问 AWS 区域中的 S3 数据。有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的创建 IAM 角色。

- 在 IAM 角色信任策略中，向 S3 访问权限管控服务 (access-grants.s3.amazonaws.com)主体授予对您创建的 IAM 角色的访问权限。为此，您可以创建一个包含以下语句的 JSON 文件。要将信任策略添加到您的账户，请参阅使用自定义信任策略创建角色。

  TestRolePolicy.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity", "sts:SetContext"],
      "Effect": "Allow",
      "Principal": {"Service":"access-grants.s3.amazonaws.com"},
   "Condition":{
  "StringEquals":{
   "aws:SourceAccount":"accountId",
   "aws:SourceArn":"arn:aws:s3:region:accountId:access-grants/default"
  }
  //Optionally, for an IAM Identity Center use case, add:
  "ForAnyValue:StringEquals": {
   "aws:RequestContextProvider":"arn:aws:iam::aws:contextProvider/IdentityCenter"
  }
 }
    }
  ]
}
```

- 创建 IAM 策略来将 Amazon S3 权限附加到您创建的 IAM 角色。请查看以下示例 iam-policy.json 文件，并将 *user input placeholders* 替换为您自己的信息。

> **Note**
>
> - 如果您使用采用 AWS Key Management Service（AWS KMS）密钥的服务器端加密来加密数据，则以下示例将在策略中包含 IAM 角色的必要 AWS KMS 权限。如果您不使用此功能，则可以从 IAM 策略中删除这些权限。
> - 仅当凭证由 S3 Access Grants 出售时，您才能将 IAM 角色限制为访问 S3 数据。此示例说明如何为特定 S3 Access Grants 实例添加 Condition 语句。要使用此 Condition，

请将 Condition 语句中的 S3 访问权限管控实例 ARN 替换为您的 S3 访问权限管控实例 ARN，其格式为：arn:aws:s3:*region*:*accountId*:access-grants/default

iam-policy.json

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
```

```
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
   //Optionally add the following section if you use SSE-KMS encryption
      {
         "Sid": "KMSPermissions",
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt",
            "kms:GenerateDataKey"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 在 S3 Access Grants 实例中注册位置。

> **Note**
>
> 在 S3 访问权限管控实例中注册第一个位置后，您的实例中仍没有任何单个访问权限管控。要创建访问权限管控，请参阅创建授权。

使用 S3 控制台

您必须先注册至少一个位置，之后才能使用 S3 Access Grants 授予对 S3 数据的访问权限。

在 S3 Access Grants 实例中注册位置

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。

   如果您是首次使用 S3 Access Grants 实例，请确保已完成步骤 1 - 创建 S3 Access Grants 实例，并导航到设置 Access Grants 实例向导的步骤 2。如果您已拥有 S3 Access Grants 实例，请选择查看详细信息，然后从位置选项卡中选择注册位置。

   a. 对于位置范围，请选择浏览 S3 或输入要注册的位置的 S3 URI 路径。有关 S3 URI 格式，请参阅位置格式表。输入 URI 后，您可以选择查看来浏览该位置。

   b. 对于 IAM 角色，请选择下列选项之一：

      - 从现有 IAM 角色中选择

        从下拉列表中选择 IAM 角色。选择一个角色后，选择查看，确保该角色具有管理要注册的位置所需的权限。具体而言，请确保此角色向 S3 Access Grants 授予 `sts:AssumeRole` 和 `sts:SetSourceIdentity` 权限。

      - 输入 IAM 角色 ARN

        导航到 IAM 控制台。复制 IAM 角色的 Amazon 资源名称（ARN），并将它粘贴到此框中。

   c. 要完成此操作，请选择下一步或注册位置。

4. 故障排除：

无法注册位置

- 该位置可能已被注册。

  您可能没有 s3:CreateAccessGrantsLocation 权限，无法注册位置。请联系您的账户管理员。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

您可以在 S3 Access Grants 实例中注册默认位置 s3:// 或自定义位置。请确保先创建一个具有该位置的主体访问权限的 IAM 角色，然后确保授予 S3 Access Grants 权限以代入此角色。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example 创建资源策略

创建允许 S3 Access Grants 代入 IAM 角色的策略。为此，您可以创建一个包含以下语句的 JSON 文件。要将资源策略添加到您的账户，请参阅创建和附加您的第一个客户管理型策略。

TestRolePolicy.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity"],
      "Effect": "Allow",
      "Principal": {"Service":"access-grants.s3.amazonaws.com"}
    }
  ]
}
```

Example 创建角色

运行以下 IAM 命令以创建该角色。

```
aws iam create-role --role-name accessGrantsTestRole \
 --region us-east-2 \
```

```
 --assume-role-policy-document file://TestRolePolicy.json
```

运行 `create-role` 命令将返回策略：

```
{
    "Role": {
        "Path": "/",
        "RoleName": "accessGrantsTestRole",
        "RoleId": "AROASRDGX4WM4GH55GIDA",
        "Arn": "arn:aws:iam::111122223333:role/accessGrantsTestRole",
        "CreateDate": "2023-05-31T18:11:06+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Stmt1685556427189",
                    "Action": [
                        "sts:AssumeRole",
                        "sts:SetSourceIdentity"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service":"access-grants.s3.amazonaws.com"
                    }
                }
            ]
        }
    }
}
```

Example

创建一个 IAM 策略，将 Amazon S3 权限附加到 IAM 角色。请查看以下示例 `iam-policy.json` 文件，并将 *`user input placeholders`* 替换为您自己的信息。

> **Note**
>
> 如果您使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密来加密数据，则以下示例将在策略中添加 IAM 角色的必要 AWS KMS 权限。如果您不使用此功能，则可以从 IAM 策略中删除这些权限。
>
> 为了确保在凭证由 S3 Access Grants 分发时，IAM 角色只能用于访问 S3 中的数据，此示例向您说明了如何在 IAM 策略中添加一个指定 S3 Access Grants 实例

（s3:AccessGrantsInstance: *InstanceArn*）的 Condition 语句。在使用以下示例策略时，请将 *user input placeholders* 替换为您自己的信息。

iam-policy.json

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
```

```
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS ##:accountId:access-
grants/default"]
            }
         }
      },
      {
         "Sid": "KMSPermissions",
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt",
            "kms:GenerateDataKey"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Example

运行以下命令：

```
aws iam put-role-policy \
--role-name accessGrantsTestRole \
--policy-name accessGrantsTestRole \
--policy-document file://iam-policy.json
```

Example 注册默认位置

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3:// \
 --iam-role-arn arn:aws:iam::111122223333:role/accessGrantsTestRole
```

响应：

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://”
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example 注册自定义位置

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3://DOC-BUCKET-EXAMPLE/ \
 --iam-role-arn arn:aws:iam::123456789012:role/accessGrantsTestRole
```

响应：

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-BUCKET-EXAMPLE/",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

使用 REST API

有关用于管理 S3 Access Grants 实例的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- CreateAccessGrantsLocation
- DeleteAccessGrantsLocation
- GetAccessGrantsLocation
- ListAccessGrantsLocations
- UpdateAccessGrantsLocation

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 注册位置。

要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

Java

您可以在 S3 Access Grants 实例中注册默认位置 `s3://` 或自定义位置。请确保先创建一个具有该位置的主体访问权限的 IAM 角色，然后确保授予 S3 Access Grants 权限以代入此角色。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example 注册默认位置

请求:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://")
.iamRoleArn("arn:aws:iam::123456789012:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

响应:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

Example 注册自定义位置

请求:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://DOC-BUCKET-EXAMPLE/")
.iamRoleArn("arn:aws:iam::111122223333:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

响应:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=18cfe6fb-eb5a-4ac5-aba9-8d79f04c2012,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/18cfe6fb-eb5a-4ac5-aba9-8d79f04c2666,
LocationScope= s3://test-bucket-access-grants-user123/,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

主题

- 查看已注册位置的详细信息

- 更新已注册位置
- 删除已注册位置

查看已注册位置的详细信息

您可以获取使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 在 S3 Access Grants 实例中注册的位置的详细信息。

使用 S3 控制台

查看在 S3 Access Grants 实例中注册的位置

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。
4. 对于实例，选择查看详细信息。
5. 在实例的详细信息页面上，选择位置选项卡。
6. 找到要查看的已注册位置。要筛选已注册位置的列表，请使用搜索框。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 获取已注册位置的详细信息

```
aws s3control get-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id default
```

响应：

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
```

```
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example – 列出在 S3 Access Grants 实例中注册的所有位置

要将结果限制为 S3 前缀或存储桶，可以选择使用 `--location-scope s3://bucket-and-or-prefix` 参数。

```
aws s3control list-access-grants-locations \
--account-id 111122223333 \
--region us-east-2
```

响应：

```
{"AccessGrantsLocationsList": [
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://"
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     },
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://amzn-s3-demo-bucket/prefixA*",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     }
   ]
  }
```

使用 REST API

有关用于获取已注册位置的详细信息或列出在 S3 Access Grants 实例中注册的所有位置的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- GetAccessGrantsLocation

- ListAccessGrantsLocations

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 获取已注册位置的详细信息，或列出 S3 Access Grants 实例中的所有已注册位置。

要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

Java

Example – 获取已注册位置的详细信息

```
public void getAccessGrantsLocation() {
GetAccessGrantsLocationRequest getAccessGrantsLocationRequest =
 GetAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("default")
.build();
GetAccessGrantsLocationResponse getAccessGrantsLocationResponse =
 s3Control.getAccessGrantsLocation(getAccessGrantsLocationRequest);
LOGGER.info("GetAccessGrantsLocationResponse: " + getAccessGrantsLocationResponse);
}
```

响应：

```
GetAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope= s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

Example – 列出 S3 Access Grants 实例中的所有已注册位置

要将结果限制为 S3 前缀或存储桶，可以选择在 `LocationScope` 参数中传递 S3 URI，例如 `s3://bucket-and-or-prefix`。

```
public void listAccessGrantsLocations() {
```

```
ListAccessGrantsLocationsRequest listRequest =
 ListAccessGrantsLocationsRequest.builder()
.accountId("111122223333")
.build();

ListAccessGrantsLocationsResponse listResponse =
 s3Control.listAccessGrantsLocations(listRequest);
LOGGER.info("ListAccessGrantsLocationsResponse: " + listResponse);
}
```

响应：

```
ListAccessGrantsLocationsResponse(
AccessGrantsLocationsList=[
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
),
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb456,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://amzn-s3-demo-bucket/prefixA*,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
]
)
```

更新已注册位置

您可以更新已在 Amazon S3 Access Grants 实例中注册的位置的 AWS Identity and Access Management（IAM）角色。对于用于在 S3 Access Grants 中注册位置的每个新的 IAM 角色，请务必向 S3 Access Grants 服务主体（`access-grants.s3.amazonaws.com`）授予该角色的访问权限。为此，请在首次注册位置时使用的同一个信任策略 JSON 文件中为新 IAM 角色添加一个条目。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 在 S3 Access Grants 实例中更新位置。

使用 S3 控制台

更新在 S3 Access Grants 实例中注册的位置的 IAM 角色

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。
4. 对于实例，选择查看详细信息。
5. 在实例的详细信息页面上，选择位置选项卡。
6. 找到要更新的位置。要筛选位置列表，请使用搜索框。
7. 选择要更新的已注册位置旁边的选项按钮。
8. 更新 IAM 角色，然后选择保存更改。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 更新已注册位置的 IAM 角色

```
aws s3control update-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id 635f1139-1af2-4e43-8131-a4de006eb999 \
--iam-role-arn arn:aws:iam::777788889999:role/accessGrantsTestRole
```

响应：

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb999",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:777788889999:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://amzn-s3-demo-bucket/prefixB*",
    "IAMRoleArn": "arn:aws:iam::777788889999:role/accessGrantsTestRole"
}
```

使用 REST API

有关用于更新 S3 Access Grants 实例中的位置的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的 UpdateAccessGrantsLocation。

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 更新已注册位置的 IAM 角色。

要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

Java

Example – 更新已注册位置的 IAM 角色

```
public void updateAccessGrantsLocation() {
UpdateAccessGrantsLocationRequest updateRequest =
 UpdateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("635f1139-1af2-4e43-8131-a4de006eb999")
.iamRoleArn("arn:aws:iam::777788889999:role/accessGrantsTestRole")
.build();
UpdateAccessGrantsLocationResponse updateResponse =
 s3Control.updateAccessGrantsLocation(updateRequest);
LOGGER.info("UpdateAccessGrantsLocationResponse: " + updateResponse);
}
```

响应：

```
UpdateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb999,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:777788889999:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://amzn-s3-demo-bucket/prefixB*,
IAMRoleArn=arn:aws:iam::777788889999:role/accessGrantsTestRole
)
```

删除已注册位置

您可以从 Amazon S3 Access Grants 实例中删除位置注册。删除该位置会将其从 S3 Access Grants 实例取消注册。

您必须先删除与该位置关联的所有授权，之后才能从 S3 Access Grants 实例中删除位置注册。有关如何删除授权的信息，请参阅删除授权。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 从 S3 Access Grants 实例中删除位置。

使用 S3 控制台

从 S3 Access Grants 实例中删除位置注册

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。
4. 对于实例，选择查看详细信息。
5. 在实例的详细信息页面上，选择位置选项卡。
6. 找到要更新的位置。要筛选位置列表，请使用搜索框。
7. 选择要删除的已注册位置旁边的选项按钮。
8. 选择注销。
9. 这将出现一个对话框，警告您此操作无法撤消。要删除该位置，请选择取消注册。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 删除位置注册

```
aws s3control delete-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id  a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
 // No response body
```

使用 REST API

有关用于从 S3 Access Grants 实例中删除位置的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的 DeleteAccessGrantsLocation。

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 删除位置。

要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

Java

Example – 删除位置注册

```
public void deleteAccessGrantsLocation() {
DeleteAccessGrantsLocationRequest deleteRequest =
 DeleteAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantsLocationResponse deleteResponse =
 s3Control.deleteAccessGrantsLocation(deleteRequest);
LOGGER.info("DeleteAccessGrantsLocationResponse: " + deleteResponse);
}
```

响应：

```
DeleteAccessGrantsLocationResponse()
```

创建授权

S3 访问权限管控实例中的单个访问权限管控 可让特定身份 [AWS Identity and Access Management（IAM）主体或公司目录中的用户或组] 在 S3 访问权限管控实例中注册的位置内获得访问权限。位置将存储桶或前缀映射到 IAM 角色。S3 访问权限管控代入此 IAM 角色，来向被授权者提供临时凭证。

在 S3 访问权限管控实例中注册至少一个位置后，可以创建访问权限管控。

被授权者可以是 IAM 用户或角色，也可以是目录用户或组。目录用户是与 S3 访问权限管控实例关联的公司目录或外部身份源中的用户。有关更多信息，请参阅 S3 Access Grants 和公司目录身份。要从 IAM Identity Center 为特定目录用户或组创建授权，请在 IAM Identity Center 中找到 IAM Identity Center 用来标识该用户的 GUID，例如 `a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`。有关如何使用 IAM Identity Center 来查看用户信息的更多信息，请参阅《AWS IAM Identity Center 用户指南》中的 View user and group assignments。

您可以授予对存储桶、前缀或对象的访问权限。Amazon S3 中的前缀是用于整理存储桶中对象的对象密钥名称开头的字符串。这可以是任何支持的字符串，例如，存储桶中以 engineering/ 前缀开头的对象键名称。

子前缀

在授予对已注册位置的访问权限时，可以使用 Subprefix 字段将访问权限范围缩小到位置范围的子集。如果您为授权选择的已注册位置是默认 S3 路径 (s3://)，则必须缩小授权范围。无法为默认位置 (s3://) 创建访问权限管控，这将向被授权者授予访问 AWS 区域中每个存储桶的权限。相反，您必须将授权范围缩小到以下各项之一：

- 存储桶：s3://*bucket*/*
- 存储桶中的前缀：s3://*bucket*/*prefix**
- 前缀中的前缀：s3://*bucket*/*prefixA*/*prefixB**
- 对象：s3://*bucket*/*object-key-name*

如果您要在已注册位置为存储桶的情况下创建访问权限管控，则可在 Subprefix 字段中传递下列项之一来缩小授权范围：

- 存储桶中的前缀：*prefix**
- 前缀中的前缀：*prefixA/prefixB**
- 对象：/*object-key-name*

在创建授权后，Amazon S3 控制台中显示的授权范围或者 API 或 AWS Command Line Interface（AWS CLI）响应中返回的 GrantScope 是将位置路径与 Subprefix 连接后的结果。确保此连接的路径正确映射到要授予其访问权限的 S3 存储桶、前缀或对象。

> **Note**
>
> - 如果要创建的访问权限管控仅授予对一个对象的访问权限，则必须指定授权类型是针对某个对象的。要在 API 调用或 CLI 命令中执行此操作，请传递带有 Object 值的 s3PrefixType 参数。在 Amazon S3 控制台中，当您创建授权时，在选择位置后，在授权范围下，选中授权范围是一个对象复选框。
> - 如果存储桶尚不存在，则无法创建存储桶的授权。但是，您可以创建对尚不存在的前缀的授权。

- 有关您可以在 S3 访问权限管控实例中创建的最大授权数量，请参阅 S3 Access Grants 限制。

您可以使用 Amazon S3 控制台、AWS CLI、Amazon S3 REST API 和 AWS SDK 创建访问授权。

使用 S3 控制台

创建访问授权

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。

   如果您是首次使用 S3 Access Grants 实例，请确保已完成步骤 2 - 注册位置，并导航到设置 Access Grants 实例向导的步骤 3。如果您已拥有 S3 Access Grants 实例，请选择查看详细信息，然后从授权选项卡中选择创建授权。

   a. 在授权范围部分中，选择或输入已注册位置。

      如果您选择了默认 `s3://` 位置，请使用子前缀框来缩小访问授权的范围。有关更多信息，请参阅子前缀。如果您只授予一个对象的访问权限，请选择授权范围为对象。

   b. 在权限和访问下，选择权限级别，即读取和/或写入。

      然后选择被授权者类型。如果您已将公司目录添加到 IAM Identity Center，并将此 IAM Identity Center 实例与 S3 Access Grants 实例关联，则可以选择 IAM Identity Center 中的目录身份。如果您选择此选项，请从 IAM Identity Center 获取用户或组的 ID，然后在此部分中输入此 ID。

      如果被授权者类型是 IAM 用户或角色，请选择 IAM 主体。在 IAM 主体类型下，选择用户或角色。然后，在 IAM 主体用户下，从列表中进行选择或输入身份的 ID。

   c. 要创建 S3 Access Grants 授权，请选择下一步或创建授权。
4. 如果已禁用下一步或创建授权，则：

   无法创建授权

   - 您可能需要先在 S3 Access Grants 实例中注册位置。
   - 您可能没有 `s3:CreateAccessGrant` 权限，无法创建访问授权。请联系您的账户管理员。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

以下示例说明如何为 IAM 主体创建访问授权请求以及如何为公司目录用户或组创建访问授权请求。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

> **Note**
>
> 如果要创建的访问授权仅授予对一个对象的访问权限，请包括必需参数 `--s3-prefix-type Object`。

Example 为 IAM 主体创建访问授权请求

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 \
--access-grants-location-configuration S3SubPrefix=prefixB* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::123456789012:user/data-consumer-3
```

Example 创建访问授权响应

```
{"CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/grant/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "AccessGrantsLocationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://DOC-BUCKET-EXAMPLE/prefix*",
    "Permission": "READ"
}
```

为目录用户或组创建访问授权请求

要为目录用户或组创建访问授权请求，您必须先运行下列命令之一来获取目录用户或组的 GUID。

Example 获取目录用户或组的 GUID

您可以通过 IAM Identity Center 控制台或使用 AWS CLI 或 AWS SDK 来查找 IAM Identity Center 用户的 GUID。以下命令列出指定的 IAM Identity Center 实例中的用户及其名称和标识符。

```
aws identitystore list-users --identity-store-id d-1a2b3c4d1234
```

此命令列出指定的 IAM Identity Center 实例中的组。

```
aws identitystore list-groups --identity-store-id d-1a2b3c4d1234
```

Example 为目录用户或组创建访问授权

此命令类似于用来为 IAM 用户或角色创建授权的命令，只是被授权者类型为 DIRECTORY_USER 或 DIRECTORY_GROUP，并且被授权者标识符为目录用户或组的 GUID。

```
aws s3control create-access-grant \
--account-id 123456789012 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix="DOC-EXAMPLE-BUCKET/rafael/*" \
--permission READWRITE \
--grantee GranteeType=DIRECTORY_USER,GranteeIdentifier=83d43802-00b1-7054-db02-
f1d683aacba5 \
```

使用 REST API

有关用于管理访问授权的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- CreateAccessGrant
- DeleteAccessGrant
- GetAccessGrant
- ListAccessGrants

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 创建访问授权。

Java

要使用以下示例，请将 *user input placeholders* 替换为您自己的信息：

> **Note**
>
> 如果要创建的访问授权仅授予对一个对象的访问权限，请包括必需参数 `.s3PrefixType(S3PrefixType.Object)`。

Example 创建访问授权请求

```
public void createAccessGrant() {
CreateAccessGrantRequest createRequest = CreateAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa")
.permission("READ")
.accessGrantsLocationConfiguration(AccessGrantsLocationConfiguration.builder().s3SubPrefix("
.grantee(Grantee.builder().granteeType("IAM").granteeIdentifier("arn:aws:iam::111122223333:u
data-consumer-3").build())
.build();
CreateAccessGrantResponse createResponse =
 s3Control.createAccessGrant(createRequest);
LOGGER.info("CreateAccessGrantResponse: " + createResponse);
}
```

Example 创建访问授权响应

```
CreateAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
AccessGrantArn=arn:aws:s3:us-east-2:444455556666:access-grants/default/grant/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
AccessGrantsLocationId=a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
```

```
S3SubPrefix=prefixB*
),
GrantScope=s3://DOC-BUCKET-EXAMPLE/prefixB,
Permission=READ
)
```

主题

- 查看授权
- 删除授权

查看授权

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 来查看 Amazon S3 Access Grants 实例中的访问授权的详细信息。

使用 S3 控制台

查看访问授权的详细信息

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。
4. 对于实例，选择查看详细信息。
5. 在详细信息页面上，选择授权选项卡。
6. 在授权部分中，找到要查看的访问授权。要筛选授权列表，请使用搜索框。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 获取访问授权的详细信息

```
aws s3control get-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
```

响应：

```
{
    "CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "Permission": "READ",
    "AccessGrantsLocationId": "12a6710f-5af8-41f5-b035-0bc795bf1a2b",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://amzn-s3-demo-bucket/"
}
```

Example – 列出 S3 Access Grants 实例中的所有访问授权

您可以选择使用以下参数将结果限制为 S3 前缀或 AWS Identity and Access Management（IAM）身份：

- 子前缀 – `--grant-scope s3://bucket-name/prefix*`
- IAM 身份 – `--grantee-type IAM` 和 `--grantee-identifier arn:aws:iam::123456789000:role/accessGrantsConsumerRole`

```
aws s3control list-access-grants \
--account-id 111122223333
```

响应：

```
{
    "AccessGrantsList": [{"CreatedAt": "2023-06-14T17:54:46.542000+00:00",
            "AccessGrantId": "dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
```

```
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "23514a34-ea2e-4ddf-b425-d0d4bfcarda1",
            "GrantScope": "s3://amzn-s3-demo-bucket/prefixA*"
        },
        {"CreatedAt": "2023-06-24T17:54:46.542000+00:00",
            "AccessGrantId": "ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-9"
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "12414a34-ea2e-4ddf-b425-d0d4bfcacao0",
            "GrantScope": "s3://amzn-s3-demo-bucket/prefixB*"
        },

    ]
}
```

使用 REST API

您可以使用 Amazon S3 API 操作来查看访问授权的详细信息，并列出 S3 Access Grants 实例中的所有访问授权。有关用于管理访问授权的 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- GetAccessGrant
- ListAccessGrants

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 获取访问授权的详细信息。

要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

Java

Example – 获取访问授权的详细信息

```
public void getAccessGrant() {
GetAccessGrantRequest getRequest = GetAccessGrantRequest.builder()
```

```
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE22222")
.build();
GetAccessGrantResponse getResponse = s3Control.getAccessGrant(getRequest);
LOGGER.info("GetAccessGrantResponse: " + getResponse);
}
```

响应：

```
GetAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE22222,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-fd3a5086-42f7-4b34-9fad-472e2942c70e,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=12a6710f-5af8-41f5-b035-0bc795bf1a2b,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://amzn-s3-demo-bucket/
)
```

Example – 列出 S3 Access Grants 实例中的所有访问授权

您可以选择使用以下参数将结果限制为 S3 前缀或（IAM）身份：

- 范围 – GrantScope=s3://*bucket-name/prefix**
- 被授权者 – GranteeType=IAM 和 GranteeIdentifier=arn:aws:iam::*111122223333*:role/*accessGrantsConsumerRole*

```
public void listAccessGrants() {
ListAccessGrantsRequest listRequest = ListAccessGrantsRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsResponse listResponse = s3Control.listAccessGrants(listRequest);
LOGGER.info("ListAccessGrantsResponse: " + listResponse);
```

```
}
```

响应：

```
ListAccessGrantsResponse(
AccessGrantsList=[
ListAccessGrantEntry(
CreatedAt=2023-06-14T17:54:46.540z,
AccessGrantId=dd8dd089-b224-4d82-95f6-975b4185bbaa,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=23514a34-ea2e-4ddf-b425-d0d4bfcarda1,
GrantScope=s3://amzn-s3-demo-bucket/prefixA
),
ListAccessGrantEntry(
CreatedAt=2023-06-24T17:54:46.540Z,
AccessGrantId=ee8ee089-b224-4d72-85f6-975b4185a1b2,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-9
),
Permission=READ,
AccessGrantsLocationId=12414a34-ea2e-4ddf-b425-d0d4bfcacao0,
GrantScope=s3://amzn-s3-demo-bucket/prefixB*
)
]
)
```

删除授权

您可以从 Amazon S3 Access Grants 实例中删除访问授权。您无法撤消访问授权删除操作。删除访问授权后，被授权者将再也无法访问您的 Amazon S3 数据。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 删除访问授权。

使用 S3 控制台

删除访问授权

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Access Grants。
3. 在 S3 Access Grants 页面上，选择包含要使用的 S3 Access Grants 实例的区域。
4. 对于实例，选择查看详细信息。
5. 在详细信息页面上，选择授权选项卡。
6. 搜索要删除的授权。找到授权后，选择它旁边的单选按钮。
7. 选择删除。这将出现一个对话框，警告您操作无法撤消。再次选择删除以删除授权。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 删除访问授权

```
aws s3control delete-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

// No response body
```

使用 REST API

有关用于管理访问授权的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的 DeleteAccessGrant。

使用 AWS SDK

此部分中的示例说明了如何使用 AWS SDK 删除访问授权。要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

Java

Example – 删除访问授权

```
public void deleteAccessGrant() {
DeleteAccessGrantRequest deleteRequest = DeleteAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantResponse deleteResponse =
 s3Control.deleteAccessGrant(deleteRequest);
LOGGER.info("DeleteAccessGrantResponse: " + deleteResponse);
}
```

响应：

```
DeleteAccessGrantResponse()
```

通过 S3 Access Grants 请求访问 Amazon S3 数据

在使用 Amazon S3 Access Grants 创建访问授权来向 AWS Identity and Access Management（IAM）主体、公司目录身份或授权应用程序授予对您的 S3 数据的访问权限之后，您的被授权者可以请求用于访问这些数据的凭证。

当应用程序或 AWS 服务 使用 `GetDataAccess` API 操作代表被授权者向 S3 Access Grants 请求对 S3 数据的访问权限时，S3 Access Grants 首先验证您是否已向该身份授予对此数据的访问权限。然后，S3 Access Grants 使用 AssumeRole API 操作来获取临时凭证令牌并将其出售给请求方。此临时凭证令牌是 AWS Security Token Service（AWS STS）令牌。

`GetDataAccess` 请求必须包含 `target` 参数，该参数指定临时凭证适用于的 S3 数据的范围。此 `target` 范围可以是授权的范围，也可以是该范围的子集，但 `target` 范围必须在已向请求者提供的授权的范围内。请求还必须指定 `permission` 参数以指示临时凭证的权限级别，即 `READ`、`WRITE` 或 `READWRITE`。

请求者可以在其凭证请求中指定临时令牌的权限级别。通过使用 `privilege` 参数，请求者可以在授权范围边界内缩小或增大临时凭证的访问范围。`privilege` 参数的默认值为 `Default`，这意味着返回的凭证的目标范围是原始授权范围。`privilege` 的另一个可能值是 `Minimal`。如果 `target` 范围从最初的授权范围缩小，则只要 `target` 范围在授权范围内，就会解除临时凭证的范围以匹配 `target` 范围。

下表详细说明了 privilege 参数对两个授权的影响。一个授权具有范围 S3://*amzn-s3-demo-bucket1*/bob/*，其中包括 *amzn-s3-demo-bucket1* 存储桶中的整个 bob/ 前缀。另一个授权具有范围 S3://*amzn-s3-demo-bucket1*/bob/reports/*，其中仅包括 *amzn-s3-demo-bucket1* 存储桶中的 bob/reports/ 前缀。

授权范围	请求的范围	特权	返回的范围	效果
S3://*amzn-s3-demo-bucket1* /bob/*	*amzn-s3-demo-bucket1* /bob/*	Default	*amzn-s3-demo-bucket1* /bob/*	请求者有权访问 *amzn-s3-demo-bucket1* 存储桶中所有带以前缀 *bob/* 开始的密钥名称的对象。
S3://*amzn-s3-demo-bucket1* /bob/*	*amzn-s3-demo-bucket1* /bob/	Minimal	*amzn-s3-demo-bucket1* /bob/	如果前缀名称 bob/ 后面没有通配符 *，则请求者只能访问 *amzn-s3-demo-bucket1* 存储桶中名为 bob/ 的对象。此类对象并不常见。请求者无权访问任何其他对象，包括那些带以 bob/ 前缀开头的密钥名称的对象。
S3://*amzn-s3-demo-bucket1* /bob/*	*amzn-s3-demo-bucket1* /bob/images/*	Minimal	*amzn-s3-demo-bucket1* /bob/images/*	请求者有权访问 *amzn-s3-demo-bucket1* 存储桶中所有带以前缀 *bob/images/** 开始的密钥名称的对象。
S3://*amzn-s3-demo-bucket1* /bob/	*amzn-s3-demo-bucket1* /bob/reports/*	Default	*amzn-s3-demo-bucket1* /bob/reports/*	请求者有权访问 *amzn-s3-demo-bucket1* 存储桶中所有带以前缀 bob/reports 开始的密钥名称的对象，它是匹配授权的范围。

授权范围	请求的范围	特权	返回的范围	效果
repo rts/*	file. txt			
S3://*amzn-s3-d emo-bucke t1* /bob/ repo rts/*	*amzn-s3-demo-bucke t1* /bob/ repo rts/ file. txt	Minimal	*amzn-s3-demo-bucke t1* /bob/reports/ file.txt	请求者只能访问 *amzn-s3-demo-bucket1* 存储桶中具有密钥名称 bob/reports/file.t xt 的对象。请求者无权访问任何其他对象。

durationSeconds 参数设置临时凭证的持续时间（以秒为单位）。默认值为 3600 秒（1 小时），但请求者（被授权者）可以指定介于 900 秒（15 分钟）和 43200 秒（12 小时）之间的范围。如果被授权者请求的值高于此最大值，请求将失败。

> **Note**
>
> 在对临时令牌的请求中，如果位置是对象，请将请求中的 targetType 参数值设置为 Object。仅当位置为对象且权限级别为 Minimal 时需要此参数。如果位置是存储桶或前缀，则无需指定此参数。

有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 GetDataAccess。

您可以使用 AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 和 AWS SDK 请求临时凭证。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example 请求临时凭证

请求:

```
aws s3control get-data-access \
--account-id 111122223333 \
--target s3://amzn-s3-demo-bucket/prefixA* \
--permission READ \
--privilege Default \
--region us-east-2
```

响应:

```
{
"Credentials": {
 "AccessKeyId": "Example-key-id",
 "SecretAccessKey": "Example-access-key",
 "SessionToken": "Example-session-token",
"Expiration": "2023-06-14T18:56:45+00:00"},
    "MatchedGrantTarget": "s3://amzn-s3-demo-bucket/prefixA**"
}
```

使用 REST API

有关用于请求 S3 Access Grants 中的临时凭证的 Amazon S3 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的 GetDataAccess。

使用 AWS SDK

此部分中的示例说明被授权者如何使用 AWS SDK 从 S3 Access Grants 请求临时凭证。

Java

以下代码示例返回被授权者用于访问您的 S3 数据的临时凭证。要使用此代码示例，请将 *user input placeholders* 替换为您自己的信息。

Example 获取临时凭证

请求:

```
public void getDataAccess() {
GetDataAccessRequest getDataAccessRequest = GetDataAccessRequest.builder()
.accountId("111122223333")
.permission(Permission.READ)
.privilege(Privilege.MINIMAL)
.target("s3://amzn-s3-demo-bucket/prefixA*")
```

```
.build();
GetDataAccessResponse getDataAccessResponse =
 s3Control.getDataAccess(getDataAccessRequest);
LOGGER.info("GetDataAccessResponse: " + getDataAccessResponse);
}
```

响应:

```
GetDataAccessResponse(
Credentials=Credentials(
AccessKeyId="Example-access-key-id",
SecretAccessKey="Example-secret-access-key",
SessionToken="Example-session-token",
Expiration=2023-06-07T06:55:24Z
))
```

通过访问授权来访问 S3 数据

在被授权者通过其访问授权获得临时凭证后，他们可以使用这些临时凭证来调用 Amazon S3 API 操作以访问您的数据。

被授权者可以使用 AWS Command Line Interface（AWS CLI）、AWS SDK 和 Amazon S3 REST API 来访问 S3 数据。

使用 AWS CLI

在被授权者从 S3 Access Grants 获得临时凭证后，他们可以使用这些凭证设置配置文件来检索数据。

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 设置配置文件

```
aws configure set aws_access_key_id "$accessKey" --profile access-grants-consumer-access-profile
aws configure set aws_secret_access_key "$secretKey" --profile access-grants-consumer-access-profile
aws configure set aws_session_token "$sessionToken" --profile access-grants-consumer-access-profile
```

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 获取 S3 数据

被授权者可以使用 get-object AWS CLI 命令来访问数据。被授权者还可以使用 put-object、ls 和其他 S3 AWS CLI 命令。

```
aws s3api get-object \
--bucket amzn-s3-demo-bucket1 \
--key myprefix \
--region us-east-2 \
--profile access-grants-consumer-access-profile
```

使用 AWS SDK

此部分中的示例说明被授权者如何使用 AWS SDK 访问 S3 数据。

Java

有关如何使用临时凭证获取 S3 数据的示例，请参阅如何使用 AWS SDK 获取对象和 AWS SDK for Java 2.x 的 Amazon S3 代码示例。

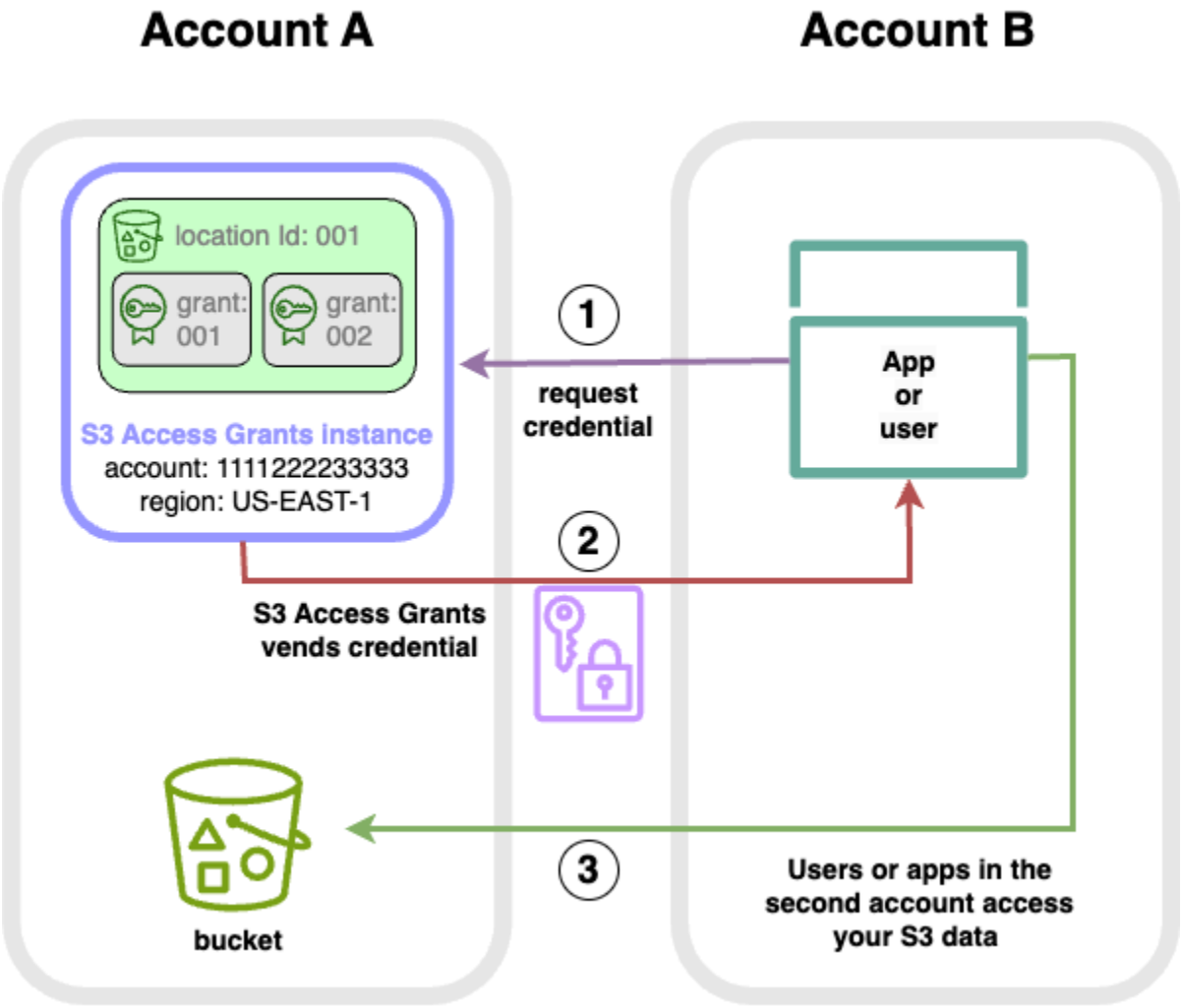
S3 Access Grants 跨账户访问

利用 S3 Access Grants，您可以授予 Amazon S3 对以下内容的数据访问权限：

- 您账户中的 AWS Identity and Access Management（IAM）身份
- 其它 AWS 账户中的 IAM 身份
- 您的 AWS IAM Identity Center 实例中的目录用户或组

首先，为另一个账户配置跨账户访问。这包括使用资源策略授予对 S3 Access Grants 实例的访问权限。然后，使用授权来授予对您的 S3 数据（存储桶、前缀或对象）的访问权限。

配置跨账户访问后，另一个账户可以从 S3 Access Grants 中请求对您的 Amazon S3 数据的临时访问凭证。下图显示了通过 S3 Access Grants 进行跨账户 S3 访问的用户流程：



1. 第二个账户（B）中的用户或应用程序向账户（A）中存储 Amazon S3 数据的 S3 Access Grants 实例请求凭证。有关更多信息，请参阅 通过 S3 Access Grants 请求访问 Amazon S3 数据。
2. 如果某个授权向第二个账户授予访问您的 Amazon S3 数据的权限，则您的账户（A）中的 S3 Access Grants 实例会返回临时凭证。有关更多信息，请参阅 the section called “创建授权”。
3. 第二个账户（B）中的用户或应用程序使用 S3 Access Grants 出售的凭证访问您的账户（A）中的 S3 数据。

配置 S3 Access Grants 跨账户访问

要通过 S3 Access Grants 授予跨账户 S3 访问权限，请执行以下步骤：

- 步骤 1：在您的账户（例如，账户 ID 111122223333）中配置在其中存储 S3 数据的 S3 Access Grants 实例。

- 步骤 2：为您的账户 111122223333 中的 S3 Access Grants 实例配置资源策略，来向第二个账户（例如，账户 ID 444455556666）授予访问权限。
- 步骤 3：为第二个账户 444455556666 中的 IAM 主体配置 IAM 权限，以便从您的账户 111122223333 中的 S3 Access Grants 实例请求凭证。
- 步骤 4：在您的账户 111122223333 中创建授权，来向第二个账户 444455556666 中的 IAM 主体授予访问您账户 111122223333 中某些 S3 数据的权限。

步骤 1：在您的账户中配置 S3 Access Grants 实例

首先，您的账户 111122223333 中必须有 S3 Access Grants 实例，才能管理对 Amazon S3 数据的访问权限。您必须在存储您要共享的 S3 数据的每个 AWS 区域中创建一个 S3 Access Grants 实例。如果您要在多个 AWS 区域中共享数据，请为每个 AWS 区域重复每个配置步骤。如果您在存储 S3 数据的 AWS 区域中已经有一个 S3 Access Grants 实例，请继续下一步。如果您尚未配置 S3 Access Grants 实例，请参阅创建 S3 访问权限管控实例来完成此步骤。

步骤 2：配置 S3 Access Grants 实例的资源策略来授予跨账户访问权限

在您的账户 111122223333 中创建 S3 Access Grants 实例来进行跨账户访问后，请为您的账户 111122223333 中的 S3 Access Grants 实例配置基于资源的策略，来授予跨账户访问权限。S3 Access Grants 实例本身支持基于资源的策略。有了正确的基于资源的策略，您就可以向其它 AWS 账户中的 AWS Identity and Access Management（IAM）用户或角色授予对 S3 Access Grants 实例的访问权限。跨账户访问仅授予以下权限（操作）：

- `s3:GetAccessGrantsInstanceForPrefix` – 用户、角色或应用程序可以检索包含特定前缀的 S3 Access Grants 实例。
- `s3:ListAccessGrants`
- `s3:ListAccessLocations`
- `s3:GetDataAccess` – 用户、角色或应用程序可以根据通过 S3 Access Grants 向您授予的访问权限请求临时凭证。使用这些凭证可访问您已获得访问权限的 S3 数据。

您可以选择将其中哪些权限包含在资源策略中。S3 Access Grants 实例上的此资源策略是基于资源的普通策略，支持 IAM 策略语言支持的所有内容。在同一个策略中，例如，您可以使用 `aws:PrincipalArn` 条件向您的账户 111122223333 中的特定 IAM 身份授予访问权限，而不必使用 S3 Access Grants 执行此操作。相反，在您的 S3 Access Grants 实例中，您可以为您账户中的单个 IAM 身份以及其它账户创建授权。通过使用 S3 Access Grants 管理每个访问授权，您可以扩展权限。

如果您已经使用 AWS Resource Access Manager（AWS RAM），您可以使用它与其它账户或在您的组织内共享您的 `s3:AccessGrants` 资源。有关更多信息，请参阅使用共享的 AWS 资源。如果您不使用 AWS RAM，也可以使用 S3 Access Grants API 操作或 AWS Command Line Interface（AWS CLI）来添加资源策略。

使用 S3 控制台

我们建议您使用 AWS Resource Access Manager（AWS RAM）控制台与其它账户或在您的组织内共享您的 `s3:AccessGrants` 资源。要跨账户共享 S3 Access Grants，请执行以下操作：

要配置 S3 Access Grants 实例资源策略，请执行以下操作：

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 从 AWS 区域选择器中选择 AWS 区域。
3. 从左侧导航窗格中，选择 Access Grants。
4. 在 Access Grants 实例页面的此账户中的实例部分，选择共享实例。这会将您重定向至 AWS RAM 控制台。
5. 选择创建资源共享。
6. 按照 AWS RAM 步骤创建资源共享。有关更多信息，请参阅 Creating a resource share in AWS RAM。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

您可以使用 `put-access-grants-instance-resource-policy` CLI 命令添加资源策略。

如果您想将您的账户 `111122223333` 中对于 S3 Access Grants 的跨账户访问权限授予第二个账户 `444455556666`，则您的账户 `111122223333` 中 S3 Access Grants 实例的资源策略应向第二个账户 `444455556666` 授予执行以下操作的权限：

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

在 S3 Access Grants 实例资源策略中，将 S3 Access Grants 实例的 ARN 指定为 `Resource`，而将第二个账户 444455556666 指定为 `Principal`。要使用以下示例，请将*#######* 替换为您自己的信息。

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

要添加或更新 S3 Access Grants 实例资源策略，可以使用以下命令。当您使用以下示例命令时，请将 *user input placeholders* 替换为您自己的信息。

Example 添加或更新 S3 Access Grants 实例资源策略

```
aws s3control put-access-grants-instance-resource-policy \
--account-id 111122223333 \
--policy file://resourcePolicy.json \
--region us-east-2
{
 "Policy": "{\n
   \"Version\": \"2012-10-17\",\n
   \"Statement\": [{\n
  \"Effect\": \"Allow\",\n
  \"Principal\": {\n
    \"AWS\": \"444455556666\"\n
  },\n
  \"Action\": [\n
    \"s3:ListAccessGrants\",\n
    \"s3:ListAccessGrantsLocations\",\n
```

```
      \"s3:GetDataAccess\",\n
      \"s3:GetAccessGrantsInstanceForPrefix\"\n
    ],\n
    \"Resource\": \"arn:aws:s3:us-east-2:111122223333:access-grants/default\"\n
      }\n
     ]\n
     }\n",
  "CreatedAt": "2023-06-16T00:07:47.473000+00:00"
 }
```

Example 获取 S3 Access Grants 资源策略

您也可以使用 CLI 来获取或删除 S3 Access Grants 实例的资源策略。

要获取 S3 Access Grants 资源策略，请使用以下示例命令。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2

{
"Policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal
\":{\"AWS\":\"arn:aws:iam::111122223333:root\"},\"Action\":[\"s3:ListAccessGrants\",
\"s3:ListAccessGrantsLocations\",\"s3:GetDataAccess\"],\"Resource\":\"arn:aws:s3:us-
east-2:111122223333:access-grants/default\"}]}",
"CreatedAt": "2023-06-16T00:07:47.473000+00:00"
}
```

Example 删除 S3 Access Grants 资源策略

要删除 S3 Access Grants 资源策略，请使用以下示例命令。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control delete-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2

// No response body
```

使用 REST API

您可以使用 PutAccessGrantsInstanceResourcePolicy API 添加资源策略。

如果您想将您的账户 111122223333 中对于 S3 Access Grants 的跨账户访问权限授予第二个账户 444455556666，则您的账户 111122223333 中 S3 Access Grants 实例的资源策略应向第二个账户 444455556666 授予执行以下操作的权限：

- s3:ListAccessGrants
- s3:ListAccessGrantsLocations
- s3:GetDataAccess
- s3:GetAccessGrantsInstanceForPrefix

在 S3 Access Grants 实例资源策略中，将 S3 Access Grants 实例的 ARN 指定为 Resource，而将第二个账户 444455556666 指定为 Principal。要使用以下示例，请将###### 替换为您自己的信息。

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

然后，您可以使用 PutAccessGrantsInstanceResourcePolicy API 来配置策略。

有关 REST API 支持更新、获取或删除 S3 Access Grants 实例的资源策略的信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- PutAccessGrantsInstanceResourcePolicy
- GetAccessGrantsInstanceResourcePolicy
- DeleteAccessGrantsInstanceResourcePolicy

使用 AWS SDK

本节为您提供 AWS SDK 示例，说明如何配置 S3 Access Grants 资源策略来向第二个 AWS 账户授予访问您的某些 S3 数据的权限。

Java

添加、更新、获取或删除资源策略以管理对 S3 Access Grants 实例的跨账户访问。

Example 添加或更新 S3 Access Grants 实例资源策略

如果您想将您的账户 111122223333 中对于 S3 Access Grants 的跨账户访问权限授予第二个账户 444455556666，则您的账户 111122223333 中 S3 Access Grants 实例的资源策略应向第二个账户 444455556666 授予执行以下操作的权限：

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

在 S3 Access Grants 实例资源策略中，将 S3 Access Grants 实例的 ARN 指定为 `Resource`，而将第二个账户 444455556666 指定为 `Principal`。要使用以下示例，请将*#######* 替换为您自己的信息。

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
```

```
}
```

要添加或更新 S3 Access Grants 实例资源策略，请使用以下代码示例：

```
public void putAccessGrantsInstanceResourcePolicy() {
 PutAccessGrantsInstanceResourcePolicyRequest putRequest =
 PutAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .policy(RESOURCE_POLICY)
 .build();
 PutAccessGrantsInstanceResourcePolicyResponse putResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(putRequest);
 LOGGER.info("PutAccessGrantsInstanceResourcePolicyResponse: " + putResponse);
 }
```

响应：

```
PutAccessGrantsInstanceResourcePolicyResponse(
 Policy={
 "Version": "2012-10-17",
 "Statement": [{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
 },
 "Action": [
 "s3:ListAccessGrants",
 "s3:ListAccessGrantsLocations",
 "s3:GetDataAccess",
 "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
 }]
 }
 )
```

Example 获取 S3 Access Grants 资源策略

要获取 S3 Access Grants 资源策略，请使用以下代码示例。要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
public void getAccessGrantsInstanceResourcePolicy() {
```

```
 GetAccessGrantsInstanceResourcePolicyRequest getRequest =
 GetAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 GetAccessGrantsInstanceResourcePolicyResponse getResponse =
 s3Control.getAccessGrantsInstanceResourcePolicy(getRequest);
 LOGGER.info("GetAccessGrantsInstanceResourcePolicyResponse: " + getResponse);
 }
```

响应：

```
GetAccessGrantsInstanceResourcePolicyResponse(
 Policy={"Version":"2012-10-17","Statement":[{"Effect":"Allow","Principal":
{"AWS":"arn:aws:iam::444455556666:root"},"Action":
["s3:ListAccessGrants","s3:ListAccessGrantsLocations","s3:GetDataAccess"],"Resource":"arn:aw
east-2:111122223333:access-grants/default"}]},
 CreatedAt=2023-06-15T22:54:44.319Z
 )
```

Example 删除 S3 Access Grants 资源策略

要删除 S3 Access Grants 资源策略，请使用以下代码示例。要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
public void deleteAccessGrantsInstanceResourcePolicy() {
 DeleteAccessGrantsInstanceResourcePolicyRequest deleteRequest =
 DeleteAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 DeleteAccessGrantsInstanceResourcePolicyResponse deleteResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(deleteRequest);
 LOGGER.info("DeleteAccessGrantsInstanceResourcePolicyResponse: " + deleteResponse);
 }
```

响应：

```
DeleteAccessGrantsInstanceResourcePolicyResponse()
```

步骤 3：向第二个账户中的 IAM 身份授予在您的账户中调用 S3 Access Grants 实例的权限

在 Amazon S3 数据的拥有者为账户 111122223333 中的 S3 Access Grants 实例配置跨账户策略后，第二个账户 444455556666 的拥有者必须为其 IAM 用户或角色创建基于身份的策略，并且拥有者必须向他们授予访问 S3 Access Grants 实例的权限。在基于身份的策略中，包括以下一项或多项操作，具体取决于在 S3 Access Grants 实例资源策略中授予的权限以及您要授予的权限：

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

按照 AWS 跨账户访问模式，第二个账户 444455556666 中的 IAM 用户或角色必须显式拥有此类权限中的一个或多个。例如，授予 `s3:GetDataAccess` 权限，以便 IAM 用户或角色可以在账户 111122223333 中调用 S3 Access Grants 实例来请求凭证。

要使用此示例命令，请将 *`user input placeholders`* 替换为您自己的信息。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "s3:GetDataAccess",
  ],
   "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
  }
 ]
}
```

有关编辑基于 IAM 身份的策略的信息，请参阅《AWS Identity and Access Management 指南》中的 Editing IAM policies。

步骤 4：在您的账户的 S3 Access Grants 实例中创建授权，来向第二个账户中的 IAM 身份授予访问您的某些 S3 数据的权限

对于最后的配置步骤，您可以在您账户 111122223333 中的 S3 Access Grants 实例中创建一个授权，来向第二个账户 444455556666 中的 IAM 身份授予访问您的账户中某些 S3 数据的权限。您可以使用 Amazon S3 控制台、CLI、API 和 SDK 实现此目的。有关更多信息，请参阅 创建授权。

在授权中，指定第二个账户中 IAM 身份的 AWS ARN，并指定 S3 数据中您要授予访问权限的具体位置（存储桶、前缀或对象）。此位置必须已在 S3 Access Grants 实例中注册。有关更多信息，请参阅注册位置。您可以选择指定子前缀。例如，如果您授予访问权限的位置是存储桶，并且您想进一步限制对该存储桶中特定对象的访问权限，则在 S3SubPrefix 字段中传递对象键名称。或者，如果您想使用以特定前缀（例如 2024-03-research-results/）开头的键名称来限制对存储桶中对象的访问权限，则传递 S3SubPrefix=2024-03-research-results/。

以下是用于为第二个账户中的身份创建访问授权的 CLI 命令示例。请参阅创建授权了解更多信息。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix=prefixA* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::444455556666:role/data-consumer-1
```

配置跨账户访问后，第二个账户中的用户或角色可以执行以下操作：

- 通过 AWS RAM 调用 ListAccessGrantsInstances 来列出与其共享的 S3 Access Grants 实例。有关更多信息，请参阅 查看 S3 Access Grants 实例的详细信息。
- 从 S3 Access Grants 请求临时凭证。有关如何发出这些请求的更多信息，请参阅通过 S3 Access Grants 请求访问 Amazon S3 数据。

将 AWS 标签与 S3 Access Grants 配合使用

Amazon S3 Access Grants 中的标签与 Amazon S3 中的对象标签具有相似的特征。每个标签都是一个键-值对。您可以在 S3 Access Grants 中标记的资源是 S3 Access Grants 实例、位置和授权。

> **Note**
>
> S3 Access Grants 中的标记使用与对象标记不同的 API 操作。S3 Access Grants 使用 TagResource、UntagResource 和 ListTagsForResource API 操作，其中资源可以是 S3 Access Grants 实例、已注册位置或访问授权。

与对象标签类似，以下限制将适用：

- 可以在创建新的 S3 Access Grants 资源时向其添加标签，也可以向现有资源添加标签。
- 您最多可以将 10 个标签与一个资源关联。如果多个标签与同一资源关联，则它们必须具有唯一的标签键。
- 标签键的长度最大可以为 128 个 Unicode 字符，标签值的长度最大可以为 256 个 Unicode 字符。标签在内部以 UTF-16 表示。采用 UTF-16 时，字符占用 1 或 2 个字符位置。
- 键和值区分大小写。

有关标签限制的更多信息，请参阅《AWS Billing 用户指南》中的用户定义的标签限制。

您可以使用 AWS Command Line Interface（AWS CLI）、Amazon S3 REST API 或 AWS SDK 在 S3 Access Grants 中标记资源。

使用 AWS CLI

要安装 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS CLI。

可以在创建 S3 Access Grants 资源时或之后为其添加标签。以下示例说明如何标记或取消标记 S3 Access Grants 实例。可以为已注册位置和访问授权执行类似的操作。

要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

Example – 创建带标签的 S3 Access Grants 实例

```
aws s3control create-access-grants-instance \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region us-east-2 \
 --tags Key=tagKey1,Value=tagValue1
```

响应：

```
 {
    "CreatedAt": "2023-10-25T01:09:46.719000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

Example – 标记已创建的 S3 Access Grants 实例

```
aws s3control tag-resource \
```

```
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2 \
--tags Key=tagKey2,Value=tagValue2
```

Example – 列出 S3 Access Grants 实例的标签

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2
```

响应：

```
{
    "Tags": [
        {
            "Key": "tagKey1",
            "Value": "tagValue1"
        },
        {
            "Key": "tagKey2",
            "Value": "tagValue2"
        }
    ]
}
```

Example – 取消标记 S3 Access Grants 实例

```
aws s3control untag-resource \
 --account-id 111122223333 \
 --resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
 --profile access-grants-profile \
 --region us-east-2 \
 --tag-keys "tagKey2"
```

使用 REST API

可以使用 Amazon S3 API 来标记、取消标记 S3 Access Grants 实例、已注册位置或访问授权，或列出这些项的标签。有关用于管理 S3 Access Grants 标签的 REST API 支持的信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- TagResource
- UntagResource
- ListTagsForResource

S3 Access Grants 限制

S3 Access Grants 具有以下限制：

> **Note**
>
> 如果您的使用场景超出了这些限制，请联系 AWS 支持以请求提高限制。

S3 Access Grants 实例

您的每个账户在每个 AWS 区域中可创建 1 个 S3 Access Grants 实例。请参阅创建 S3 Access Grants 实例。

S3 Access Grants 位置

您可以为每个 S3 Access Grants 实例注册 1000 个 S3 Access Grants 位置。请参阅注册 S3 Access Grants 位置。

授权

可以为每个 S3 Access Grants 实例创建 10 万个授权。请参阅创建授权。

S3 Access Grants 集成

S3 Access Grants 可与以下 AWS 服务和功能结合使用。在提供了新的集成时，此页面将更新。

Amazon Athena

使用已启用 IAM Identity Center 的 Athena 工作组

Amazon EMR

使用 S3 Access Grants 启动 Amazon EMR 集群

Amazon EMR on EKS

使用 S3 Access Grants 启动 Amazon EMR on EKS 集群

Amazon EMR Serverless 应用程序

使用 S3 Access Grants 启动 Amazon EMR Serverless 应用程序

AWS IAM Identity Center

跨应用程序的可信身份传播

Amazon SageMaker Studio

Amazon S3 访问权限管控现已与 Amazon SageMaker Studio 集成

开源 Python 框架

Amazon S3 访问权限管控现已与开源 Python 框架集成

使用 ACL 管理访问

访问控制列表（ACL）是基于资源的选项之一，可用于管理对存储桶和对象的访问。您可以使用 ACL 向其他 AWS 账户授予基本的读/写权限。使用 ACL 管理权限有一些限制。

例如，您可以仅向其他 AWS 账户授予权限；您无法向您账户中的用户授予权限。无法授予条件性权限，也无法显式拒绝权限。ACL 适用于特定情景。例如，如果存储桶拥有者允许其他 AWS 账户上传对象，则对这些对象的权限只能由拥有此对象的 AWS 账户使用对象 ACL 加以管理。

S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权和禁用或启用 ACL。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对这些对象的访问权限。

Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有需要单独控制每个对象的访问权限的特殊情况。禁用 ACL 后，您可以使用策略来控制对存储桶中所有对象的访问权限，无论是谁将对象上传到您的存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

> **Important**
>
> 如果您的存储桶针对 S3 对象所有权使用强制存储桶拥有者设置，则必须使用策略授予对存储桶及其中对象的访问权限。启用强制存储桶拥有者设置后，设置访问控制列表（ACL）或更新 ACL 的请求将失败并返回 `AccessControlListNotSupported` 错误代码。仍然支持读取 ACL 的请求。

有关 ACL 的更多信息，请参阅以下主题：

主题

- 访问控制列表 (ACL) 概述
- 配置 ACL
- 适用于 ACL 的策略示例

访问控制列表 (ACL) 概述

Amazon S3 访问控制列表（ACL）使您可以管理存储桶和对象的访问权限。每个存储桶和对象都有一个作为子资源而附加的 ACL。它定义了哪些 AWS 账户或组将被授予访问权限以及访问的类型。收到针对某个资源的请求后，Amazon S3 将检查相应的 ACL 以验证请求者是否拥有所需的访问权限。

S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权和禁用或启用 ACL。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对这些对象的访问权限。

Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有需要单独控制每个对象的访问权限的特殊情况。禁用 ACL 后，您可以使用策略来控制对存储桶中所有对象的访问权限，无论是谁将对象上传到您的存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

> **Important**
>
> 如果您的存储桶针对 S3 对象所有权使用强制存储桶拥有者设置，则必须使用策略授予对存储桶及其中对象的访问权限。启用强制存储桶拥有者设置后，设置访问控制列表（ACL）或更新 ACL 的请求将失败并返回 `AccessControlListNotSupported` 错误代码。仍然支持读取 ACL 的请求。

创建存储桶或对象时，Amazon S3 将创建一个默认 ACL 以授予资源拥有者对资源的完全控制权限。这显示在下面的示例存储桶 ACL 中（默认对象 ACL 具有相同的结构）：

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>*** Owner-Canonical-User-ID ***</ID>
    <DisplayName>owner-display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="Canonical User">
        <ID>*** Owner-Canonical-User-ID ***</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

示例 ACL 包含一个可通过 AWS 账户的规范用户 ID 识别拥有者的 `Owner` 元素。有关查找规范用户 ID 的说明，请参阅查找 AWS 账户规范用户 ID。`Grant` 元素将识别被授权者（AWS 账户或预定义的组）和所授予的权限。此默认的 ACL 拥有一个适用于所有者的 `Grant` 元素。您可以通过添加 `Grant` 元素授予权限，每个授权都将识别被授权者和权限。

> **Note**
>
> ACL 可以拥有最多 100 个授权。

主题

- 谁是被授权者？
- 我能授予哪些许可？
- 常见 Amazon S3 请求的 aclRequired 值
- 示例 ACL
- 标准 ACL

谁是被授权者？

被授权者可以是 AWS 账户或某个预定义的 Amazon S3 组。您可以使用电子邮件地址或规范用户 ID 向 AWS 账户授予权限。但是，如果您在授权请求中提供电子邮件地址，Amazon S3 将为该账户查找规范用户 ID 并将它添加到 ACL。生成的 ACL 始终包含 AWS 账户的规范用户 ID，而不是 AWS 账户的电子邮件地址。

授予访问权限时，可以将每个被授权者指定为一个 *type*="*value*" 对，其中 *type* 为以下值之一：

- `id` – 如果指定的值是 AWS 账户的规范用户 ID
- `uri` – 如果您正在向预定义组授予权限
- `emailAddress` – 如果指定的值是 AWS 账户的电子邮件地址

> **Important**
>
> 仅以下 AWS 区域中支持使用电子邮件地址指定被授权者：
>
> - 美国东部（弗吉尼亚北部）
> - 美国西部（加利福尼亚北部）
> - 美国西部（俄勒冈）
> - 亚太地区（新加坡）
> - 亚太地区（悉尼）
> - 亚太地区（东京）
> - 欧洲（爱尔兰）
> - 南美洲（圣保罗）
>
> 有关 Amazon S3 支持的所有区域和端点的列表，请参阅《Amazon Web Services 一般参考》中的区域和端点。

Example 示例：电子邮件地址

例如，以下 `x-amz-grant-read` 标头向由电子邮件地址标识的 AWS 账户授予读取对象数据及其元数据的权限：

```
x-amz-grant-read: emailAddress="xyz@example.com", emailAddress="abc@example.com"
```

> **Warning**
>
> 当您需要向其他 AWS 账户授权访问您的资源时，注意 AWS 账户可以向其账户下的用户授予权限。这称为跨账户访问。有关使用跨账户访问的信息，请参阅 IAM 用户指南中的创建角色以向 IAM 用户委派权限。

查找 AWS 账户规范用户 ID

规范用户 ID 与您的 AWS 账户关联。此 ID 是一个长串字符，例如：

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

有关如何查找您账户的规范用户 ID 的信息，请参阅《AWS 账户管理参考指南》中的 Find the canonical user ID for your AWS 账户。

您也可以通过读取 AWS 账户 有权访问的存储桶或对象的 ACL，查找 AWS 账户 的规范用户 ID。如果某个 AWS 账户通过授权请求被授予了许可，ACL 中将新增一个授权条目和账户的规范用户 ID。

> **Note**
>
> 如果您公开存储桶（不建议），则任何未经身份验证的用户都可以将对象上传到该存储桶。这些匿名用户没有 AWS 账户。当匿名用户将对象上传到您的存储桶时，Amazon S3 会在 ACL 中添加特殊的规范用户 ID（`65a011a29cdf8ec533ec3d1ccaae921c`）作为对象拥有者。有关更多信息，请参阅 Amazon S3 存储桶和对象所有权。

Amazon S3 预定义的组

Amazon S3 拥有一系列预定义的组。将账户访问权限授予某个组时，您可以指定我们的一个 Amazon S3 URI，而不是规范用户 ID。Amazon S3 提供以下预定义组：

- “经身份验证的用户”组 – 由 `http://acs.amazonaws.com/groups/global/AuthenticatedUsers` 表示。

  该组代表了所有 AWS 账户。该组的访问权限允许任何 AWS 账户 访问资源。但是，所有的请求必须是已签名的 (经身份验证)。

> **Warning**
>
> 在您向经身份验证的用户组授予访问权限后，世界上任何经身份验证的 AWS 用户都可以访问您的资源。

- “所有用户”组 – 由 `http://acs.amazonaws.com/groups/global/AllUsers` 表示。

  授予此组的访问权限将允许世界上的任何人访问资源。请求可以是已签名的 (经身份验证)，也可以是未签名的 (匿名)。未签名的请求将省略请求中的 Authentication 标头。

> **Warning**
>
> 强烈建议您绝不要向 All Users 组授予 `WRITE`、`WRITE_ACP` 或 `FULL_CONTROL` 权限。例如，尽管 `WRITE` 权限不允许非所有者覆盖或删除现有对象，但 `WRITE` 权限仍允许任何人在您的存储桶中存储对象，而您需要为此付费。有关这些权限的详细信息，请参阅下一节 我能授予哪些许可？。

- “日志传输”组 – 由 `http://acs.amazonaws.com/groups/s3/LogDelivery` 表示。

  对于存储桶的 `WRITE` 权限使该组可以将服务器访问日志（请参阅使用服务器访问日志记录来记录请求）写入存储桶。

> **Note**
>
> 使用 ACL 时，被授权者可以是 AWS 账户或者某个预定义的 Amazon S3 组。但是，被授权者不能是 IAM 用户。有关 IAM 中 AWS 用户和权限的更多信息，请参阅使用 AWS Identity and Access Management。

我能授予哪些许可？

下表列出了 Amazon S3 在 ACL 中支持的权限集。对于对象 ACL 和存储桶 ACL，ACL 权限集相同。但是，根据上下文（存储桶 ACL 或对象 ACL），这些 ACL 权限将授予针对特定存储桶或对象操作的权限。下表列出了权限并描述这些权限在对象和存储桶上下文中的意义。

有关使用 Amazon S3 控制台 ACL 权限的更多信息，请参阅 配置 ACL。

ACL 权限

许可	在存储桶上授权的时间	在对象上授权的时间
READ	允许被授权者列出存储桶中的对象	允许被授权者读取对象数据及其元数据
WRITE	允许被授权者在存储桶中创建新对象。对于现有对象的存储桶和对象拥有者，还允许删除和覆写这些对象	不适用
READ_ACP	允许被授权者读取存储桶 ACL	允许被授权者读取对象 ACL
WRITE_ACP	允许被授权者为适用的存储桶编写 ACL	允许被授权者为适用的对象编写 ACL
FULL_CONTROL	允许被授予者对存储桶拥有 READ、WRITE、READ_ACP 和 WRITE_ACP 权限	允许被授予者对于对象拥有 READ、READ_ACP 和 WRITE_ACP 权限

> **Warning**
>
> 在授予对您的 S3 存储桶和对象的访问权限时应谨慎使用。例如，授予对存储桶的 WRITE 权限将允许被授权者创建存储桶中的任何对象。我们强烈建议您在授予权限之前通读整个访问控制列表 (ACL) 概述部分。

ACL 权限和访问策略权限的映射

如先前表中所示，相比于在访问策略中可设置的权限数目（请参阅Amazon S3 的策略操作），ACL 仅允许授予有限数量的权限。其中的每个权限允许一个或多个 Amazon S3 操作。

下表显示每个 ACL 权限如何映射到相应的访问策略权限。正如您所见，与 ACL 相比，访问策略允许的权限更多。您可以将 ACL 主要用于授予基本的读/写权限（类似于文件系统权限）。有关何时使用 ACL 的更多信息，请参阅 Amazon S3 的身份和访问管理。

有关使用 Amazon S3 控制台 ACL 权限的更多信息，请参阅 配置 ACL。

ACL 权限	当在存储桶上授予 ACL 权限时的相应访问策略	当在对象上授予 ACL 权限时的相应访问策略
READ	s3:ListBucket 、s3:ListBucketVersions 和 s3:ListBucketMultipartUploads	s3:GetObject 和 s3:GetObjectVersion
WRITE	s3:PutObject 存储桶拥有者可以创建、覆写和删除存储桶中的任何对象，而对象拥有者对其对象拥有 FULL_CONTROL 。 此外，如果被授权者是存储桶拥有者，使用存储桶 ACL 来授予 WRITE 许可将允许对该存储桶中的任意版本执行 s3:DeleteObjectVersion 操作。	不适用
READ_ACP	s3:GetBucketAcl	s3:GetObjectAcl 和 s3:GetObjectVersionAcl
WRITE_ACP	s3:PutBucketAcl	s3:PutObjectAcl 和 s3:PutObjectVersionAcl
FULL_CONTROL	等同于授予 READ、WRITE、READ_ACP 和 WRITE_ACP ACL 权限。因此，此 ACL 权限将映射到相应访问策略权限的组合。	等同于授予 READ、READ_ACP 和 WRITE_ACP ACL 权限。因此，此 ACL 权限将映射到相应访问策略权限的组合。

条件键

授予访问策略权限时，您可以使用条件键通过存储桶策略对某个对象限制 ACL 的值。以下上下文键对应于 ACL。您可以使用这些上下文键强制在请求中使用特定 ACL：

- s3:x-amz-grant-read - 需要读取访问权限。
- s3:x-amz-grant-write - 需要写入访问权限。

- `s3:x-amz-grant-read-acp` - 需要对存储桶 ACL 的读取访问权限。
- `s3:x-amz-grant-write-acp` - 需要对存储桶 ACL 的写入访问权限。
- `s3:x-amz-grant-full-control` - 需要完全控制权限。
- `s3:x-amz-acl` - 需要一个 标准 ACL。

有关涉及 ACL 特定标头的策略示例，请参阅授予 s3:PutObject 权限，指定了要求存储桶拥有者获得完全控制的条件。有关 Amazon S3 特定条件键的完整列表，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

常见 Amazon S3 请求的 **`aclRequired`** 值

要识别需要 ACL 进行授权的 Amazon S3 请求，您可以使用 Amazon S3 服务器访问日志或 AWS CloudTrail 中的 `aclRequired` 值。CloudTrail 或 Amazon S3 服务器访问日志中显示的 `aclRequired` 值取决于调用了哪些操作以及有关请求者、对象拥有者和存储桶拥有者的某些信息。如果不需要 ACL，或者您正在设置 `bucket-owner-full-control` 标准 ACL，或者如果您的存储桶策略允许请求，则 Amazon S3 服务器访问日志中的 `aclRequired` 值字符串为“-”，但 CloudTrail 中不存在该值字符串。

下表列出了 CloudTrail 或 Amazon S3 服务器访问日志中各种 Amazon S3 API 操作的预期 `aclRequired` 值。您可以使用此信息来了解哪些 Amazon S3 操作依赖于 ACL 进行授权。在下表中，A、B 和 C 代表与请求者、对象拥有者和存储桶拥有者关联的不同账户。带有星号 (*) 的条目表示账户 A、B 或 C 中的任意一个。

> **Note**
>
> 除非另有说明，否则下表中的 `PutObject` 操作表示未设置 ACL 的请求，除非该 ACL 是 `bucket-owner-full-control` ACL。`aclRequired` 为 null 值表示 AWS CloudTrail 日志中不存在 `aclRequired`。

CloudTrail 的 **`aclRequired`** 值

操作名称	请求者	对象拥有者	存储桶拥有者	存储桶策略授予访问权限	**`aclRequired`** 值	Reason
`GetObject`	A	A	A	是或否	null	同一账户的访问

操作名称	请求者	对象拥有者	存储桶拥有者	存储桶策略授予访问权限	**aclRequired** 值	Reason
	A	B	A	是或否	null	强制存储桶拥有者的同账户访问
	A	A	B	是	null	存储桶策略授予的跨账户访问
	A	A	B	否	是	跨账户访问依赖于 ACL
	A	A	B	是	null	存储桶策略授予的跨账户访问
	A	B	B	否	是	跨账户访问依赖于 ACL
	A	B	C	是	null	存储桶策略授予的跨账户访问
	A	B	C	否	是	跨账户访问依赖于 ACL
PutObject	A	不适用	A	是或否	null	同一账户的访问
	A	不适用	B	是	null	存储桶策略授予的跨账户访问

操作名称	请求者	对象拥有者	存储桶拥有者	存储桶策略授予访问权限	**aclRequired** 值	Reason
	A	不适用	B	否	是	跨账户访问依赖于 ACL
使用 ACL 的 PutObject （bucket-owner-full-control 除外）	*	不适用	*	是或否	是	请求授予 ACL
ListObjects	A	不适用	A	是或否	null	同一账户的访问
	A	不适用	B	是	null	存储桶策略授予的跨账户访问
	A	不适用	B	否	是	跨账户访问依赖于 ACL
DeleteObject	A	不适用	A	是或否	null	同一账户的访问
	A	不适用	B	是	null	存储桶策略授予的跨账户访问
	A	不适用	B	否	是	跨账户访问依赖于 ACL

操作名称	请求者	对象拥有者	存储桶拥有者	存储桶策略授予访问权限	**aclRequired** 值	Reason
PutObjectAcl	*	*	*	是或否	是	请求授予 ACL
PutBucketAcl	*	不适用	*	是或否	是	请求授予 ACL

> **Note**
>
> 除非另有说明，否则下表中的 REST.PUT.OBJECT 操作表示未设置 ACL 的请求，除非该 ACL 是 bucket-owner-full-control ACL。aclRequired 值字符串为“-”表示 Amazon S3 服务器访问日志中的 null 值。

Amazon S3 服务器访问日志的 **aclRequired** 值

操作名称	请求者	对象拥有者	存储桶拥有者	存储桶策略授予访问权限	**aclRequired** 值	Reason
REST.GET.OBJECT	A	A	A	是或否	-	同一账户的访问
	A	B	A	是或否	-	强制存储桶拥有者的同账户访问
	A	A	B	是	-	存储桶策略授予的跨账户访问
	A	A	B	否	是	跨账户访问依赖于 ACL

操作名称	请求者	对象拥有者	存储桶拥有者	存储桶策略授予访问权限	**aclRequired** 值	Reason
	A	B	B	是	-	存储桶策略授予的跨账户访问
	A	B	B	否	是	跨账户访问依赖于 ACL
	A	B	C	是	-	存储桶策略授予的跨账户访问
	A	B	C	否	是	跨账户访问依赖于 ACL
REST.PUT.OBJECT	A	不适用	A	是或否	-	同一账户的访问
	A	不适用	B	是	-	存储桶策略授予的跨账户访问
	A	不适用	B	否	是	跨账户访问依赖于 ACL

操作名称	请求者	对象拥有者	存储桶拥有者	存储桶策略授予访问权限	**aclRequired** 值	Reason
使用 ACL 的 REST.PUT.OBJECT （bowner-full-control 除外）	*	不适用	*	是或否	是	请求授予 ACL
REST.GET.BUCKET	A	不适用	A	是或否	-	同一账户的访问
	A	不适用	B	是	-	存储桶策略授予的跨账户访问
	A	不适用	B	否	是	跨账户访问依赖于 ACL
REST.DELETE.OBJECT	A	不适用	A	是或否	-	同一账户的访问
	A	不适用	B	是	-	存储桶策略授予的跨账户访问
	A	不适用	B	否	是	跨账户访问依赖于 ACL
REST.PUT.ACL	*	*	*	是或否	是	请求授予 ACL

示例 ACL

下面的存储桶上的示例 ACL 可识别资源所有者和一系列的授权。格式是指 Amazon S3 REST API 中的 ACL 的 XML 表示形式。存储桶拥有者拥有资源的 FULL_CONTROL。此外，ACL 还演示了如何将对于某个资源的权限授予两个 AWS 账户（由规范用户 ID 标识）以及上一节讨论的两个预定义 Amazon S3 组。

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>Owner-canonical-user-ID</ID>
    <DisplayName>display-name</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>Owner-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user1-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user2-canonical-user-ID</ID>
        <DisplayName>display-name</DisplayName>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
```

```
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

  </AccessControlList>
</AccessControlPolicy>
```

标准 ACL

Amazon S3 支持一系列预定义的授权，称为标准 ACL。每个标准 ACL 都有一组预定义的被授权者和许可。下表列出了一系列标准 ACL 和相关联的预定义授权。

标准 ACL	适用于	添加到 ACL 的权限
`private`	存储桶和对象	所有者将获得 `FULL_CONTROL` 。其他人没有访问权限 (默认)。
`public-read`	存储桶和对象	所有者将获得 `FULL_CONTROL` 。`AllUsers` 组 (参阅 谁是被授权者？) 将获得 `READ` 访问权限。
`public-read-write`	存储桶和对象	所有者将获得 `FULL_CONTROL` 。`AllUsers` 组将获得 `READ` 和 `WRITE` 访问权限。通常不建议在存储桶上授予该权限。
`aws-exec-read`	存储桶和对象	所有者将获得 `FULL_CONTROL` 。Amazon EC2 从 Amazon S3 获取对 `READ` Amazon Machine Image (AMI) 服务包的 `GET` 访问权限。
`authenticated-read`	存储桶和对象	所有者将获得 `FULL_CONTROL` 。`AuthenticatedUsers` 组将获得 `READ` 访问权限。

标准 ACL	适用于	添加到 ACL 的权限
bucket-owner-read	对象	对象所有者将获得 FULL_CONTROL 。存储桶拥有者将获得 READ 访问权限。如果您在创建存储段时指定此标准的 ACL，Amazon S3 将忽略它。
bucket-owner-full-control	对象	对象所有者和存储桶拥有者均可获得对对象的 FULL_CONTROL 。如果您在创建存储段时指定此标准的 ACL，Amazon S3 将忽略它。
log-delivery-write	存储桶	LogDelivery 组将获得针对存储桶的 WRITE 和 READ_ACP 许可。有关日志的更多信息，请参阅（使用服务器访问日志记录来记录请求）。

> **Note**
>
> 您可以在请求中仅指定这些标准 ACL 中的一个。

您可以使用 x-amz-acl 请求标头在请求中指定标准 ACL。当 Amazon S3 收到包含标准 ACL 的请求时，它会向资源的 ACL 添加预定义的授权。

配置 ACL

本节介绍如何使用访问控制列表（ACL）管理 S3 存储桶和对象的访问权限。您可以使用 AWS Management Console、AWS Command Line Interface（CLI)、REST API 或 AWS SDK 向资源 ACL 添加授权。

存储桶和对象的权限是相互独立的。对象不继承其存储桶的权限。例如，如果您创建了一个存储桶并授予一个用户写入权限，则将无法访问此用户的对象，除非此用户显式授予您访问权限。

您可以向其他 AWS 账户用户或预定义的组授予权限。您将向其授予权限的用户或组称作被授权者。默认情况下，拥有者（即创建存储桶的 AWS 账户）具有完全权限。

您为用户或组授予的每项权限都会在 ACL 中添加一个与该存储桶关联的条目。ACL 列出了授权，这些授权确定了被授权者和授予的权限。

S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权和禁用或启用 ACL。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于

禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对这些对象的访问权限。

Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有需要单独控制每个对象的访问权限的特殊情况。禁用 ACL 后，您可以使用策略来控制对存储桶中所有对象的访问权限，无论是谁将对象上传到您的存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

> **Important**
>
> 如果您的存储桶针对 S3 对象所有权使用强制存储桶拥有者设置，则必须使用策略授予对存储桶及其中对象的访问权限。启用强制存储桶拥有者设置后，设置访问控制列表（ACL）或更新 ACL 的请求将失败并返回 `AccessControlListNotSupported` 错误代码。仍然支持读取 ACL 的请求。

> **Warning**
>
> 我们强烈建议您避免向所有人（公有访问）或经过身份验证的用户组（所有经 AWS 身份验证的用户）群组授予写入访问权限。有关向这些组授予写访问权限的效果的更多信息，请参阅 Amazon S3 预定义的组。

使用 S3 控制台为存储桶设置 ACL 权限

控制台显示重复被授权者的组合访问授权。要查看 ACL 的完整列表，请使用 Amazon S3 REST API、AWS CLI 或 AWS SDK。

下表显示了您可以在 Amazon S3 控制台中为存储桶配置的 ACL 权限。

存储桶的 Amazon S3 控制台 ACL 权限

控制台权限	ACL 权限	访问
Objects（对象）– List（列表）	`READ`	允许被授权者列出存储桶中的对象

控制台权限	ACL 权限	访问
Objects（对象）— Write（写）	WRITE	允许被授权者在存储桶中创建新对象。对于现有对象的存储桶和对象所有者，还允许删除和覆写这些对象。
Bucket ACL（存储桶 ACL）– Read（读取）	READ_ACP	允许被授权者读取存储桶 ACL
BucketACL（存储桶 ACL）— Write（写入）	WRITE_ACP	允许被授权者为适用的存储桶编写 ACL
所有人（公共访问）：Objects（对象）— List（列表）	READ	授予存储桶中对象的公共读取访问权限。当您向所有人授予列表访问权限（公共读取权限）时，则任何人都可以访问存储桶中的对象。
所有人（公共访问）：Bucket ACL（存储桶 ACL）— Read（读取）	READ_ACP	授予存储桶 ACL 公共读取访问权限。当您向所有人授予读取访问权限（公共访问权限）时，则任何人都可以访问存储桶 ACL。

有关 ACL 权限的更多信息，请参阅 访问控制列表 (ACL) 概述。

> **Important**
>
> 如果您的存储桶针对 S3 对象所有权使用强制存储桶拥有者设置，则必须使用策略授予对存储桶及其中对象的访问权限。启用强制存储桶拥有者设置后，设置访问控制列表（ACL）或更新 ACL 的请求将失败并返回 `AccessControlListNotSupported` 错误代码。仍然支持读取 ACL 的请求。

为存储桶设置 ACL 权限

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要为其配置权限的存储桶的名称。
3. 选择 Permissions (权限)。
4. 在 Access control list (访问控制列表) 下，请选择 Edit (编辑)。

   您可以编辑存储桶的以下 ACL 权限：

   对象

   - 列出 – 允许被授权者列出存储桶中的对象。
   - 写入 – 允许被授权者在存储桶中创建新对象。对于现有对象的存储桶和对象所有者，还允许删除和覆写这些对象。

     在 S3 控制台中，您只能向 S3 日志传输组和存储桶拥有者（您的 AWS 账户）授予写入访问权限。我们强烈建议您不要向其他被授权者授予写入访问权限。但是，如果您需要授予写入访问权限，可以使用 AWS CLI、AWS SDK 或 REST API。

   Bucket ACL

   - 读取 – 允许被授权者读取存储桶 ACL。
   - 写入 – 允许被授权者为适用的存储桶编写 ACL
5. 要更改存储桶拥有者的权限，请在 Bucket owner（your AWS 账户）（存储桶拥有者 [您的 Amazon 账户]）旁清除或从以下 ACL 权限中选择：

   - Objects（对象）– List（列出）或 Write（写入）
   - Bucket ACL（存储桶 ACL）– Read（读取）或 Write（写入）

拥有者是指 AWS 账户根用户，而不是 AWS Identity and Access Management IAM 用户。有关根用户的更多信息，请参阅《IAM 用户指南》中的 AWS 账户根用户。

6. 要向公众（互联网上的所有人）授予或撤消权限，请在 每个人（公共访问）旁清除或从以下 ACL 权限中选择：

   - Objects（对象）– List（列表）
   - Bucket ACL（存储桶 ACL）– Read（读取）

   > **Warning**
   >
   > 在授予 Everyone 组对您的 S3 存储桶的公有访问权限时应谨慎使用。如果您向此组授予访问权限，那么世界上的任何人都可以访问您的存储桶。我们强烈建议您绝对不要授予对 S3 存储桶的任何类型的公有写入权限。

7. 要向拥有 AWS 账户的任何人授予或撤消权限，请在 Authenticated Users group (anyone with an AWS 账户) (已经过身份验证的用户组 [拥有 Amazon 账户的任何人]) 旁清除或从以下 ACL 权限中进行选择：

   - Objects（对象）– List（列表）
   - Bucket ACL（存储桶 ACL）– Read（读取）

8. 要授予或撤消 Amazon S3 将服务器访问日志写入存储桶的权限，请在 S3 log delivery group（S3 日志传输组）下清除或从以下 ACL 权限中进行选择：

   - Objects（对象）– List（列出）或 Write（写入）
   - Bucket ACL（存储桶 ACL）– Read（读取）或 Write（写入）

   如果存储桶设置为目标存储桶以接收访问日志，则存储桶权限必须允许日志传输组对存储桶有写入权限。当您在存储桶上启用服务器访问日志记录时，Amazon S3 控制台会向 Log Delivery（日志传输）组授予对您选择用来接收日志的目标存储桶的写入权限。有关服务器访问日志记录的更多信息，请参阅 启用 Amazon S3 服务器访问日志记录。

9. 要授予对其他 AWS 账户 的访问权限，请执行以下操作：

   a. 请选择 Add grantee（添加被授权者）。

   b. 在 Grantee (被授权者) 框中，输入其他 AWS 账户的规范 ID。

   c. 从以下 ACL 权限中进行选择：

- Objects（对象）– List（列出）或 Write（写入）
- Bucket ACL（存储桶 ACL）– Read（读取）或 Write（写入）

> **Warning**
>
> 当您需要向其他 AWS 账户授权访问您的资源时，注意 AWS 账户可以向其账户下的用户授予权限。这称为跨账户访问。有关使用跨账户访问的信息，请参阅 IAM 用户指南中的创建角色以向 IAM 用户委派权限。

10. 要删除对其他 AWS 账户的访问权限，请在 Access for other AWS 账户 (其他 Amazon 账户的访问权限) 下，请选择 Remove (删除)。
11. 要保存您的更改，请选择 Save Changes (保存更改)。

使用 S3 控制台为对象设置 ACL 权限

控制台显示重复被授权者的组合访问授权。要查看 ACL 的完整列表，请使用 Amazon S3 REST API、AWS CLI 或 AWS SDK。下表显示了您可以在 Amazon S3 控制台中为对象配置的 ACL 权限。

对象的 Amazon S3 控制台 ACL 权限

控制台权限	ACL 权限	访问
Object（对象）— Read（读取）	`READ`	允许被授权者读取对象数据及其元数据
Object ACL（对象 ACL）— Read（读取）	`READ_ACP`	允许被授权者读取对象 ACL
Object ACL（对象 ACL）—	`WRITE_ACP`	允许被授权者为适用的对象编写 ACL

控制台权限	ACL 权限	访问
Write（写入）		

有关 ACL 权限的更多信息，请参阅 访问控制列表 (ACL) 概述。

> **Important**
>
> 如果您的存储桶针对 S3 对象所有权使用强制存储桶拥有者设置，则必须使用策略授予对存储桶及其中对象的访问权限。启用强制存储桶拥有者设置后，设置访问控制列表（ACL）或更新 ACL 的请求将失败并返回 `AccessControlListNotSupported` 错误代码。仍然支持读取 ACL 的请求。

为对象设置 ACL 权限

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket（存储桶）列表中，请选择包含对象的存储桶的名称。
3. 在 objects（对象）列表中，请选择要为其设置权限的对象的名称。
4. 请选择 Permissions。
5. 在访问控制列表 (ACL) 下，请选择 Edit (编辑)。

   您可以编辑对象的以下 ACL 权限：

   对象

   - 读取 – 允许被授权者读取对象数据及其元数据。

   对象 ACL

   - 读取 – 允许被授权者读取对象 ACL。
   - 写入 – 允许被授权者为适用的对象编写 ACL。在 S3 控制台中，您只能向存储桶拥有者（您的 AWS 账户）授予写入访问权限。我们强烈建议您不要向其他被授权者授予写入访问权限。但是，如果您需要授予写入访问权限，可以使用 AWS CLI、AWS SDK 或 REST API。
6. 您可以管理对象的以下访问权限：

a. 对象所有者的访问权限

拥有者是指 AWS 账户根用户，而不是 AWS Identity and Access Management IAM 用户。有关根用户的更多信息，请参阅《IAM 用户指南》中的 AWS 账户根用户。

要更改拥有者的对象访问权限，请在 Access for object owner (对象所有者的访问权限) 下，请选择 Your AWS Account (owner) (您的 Amazon 账户 [所有者])。

选中要更改的权限所对应的复选框，然后选择 Save (保存)。

b. 其他 AWS 账户的访问权限

要向其他 AWS 账户 的 AWS 用户授予权限，请在 Access for other AWS 账户 (其他 Amazon 账户的访问权限) 下面，请选择 Add account (添加账户)。在 Enter an ID (输入 ID) 字段中，输入要向其授予对象权限的 AWS 用户的规范 ID。有关查找规范 ID 的信息，请参阅《Amazon Web Services 一般参考》中的您的 AWS 账户标识符。您可以添加多达 99 个用户。

选中要向用户授予的权限所对应的复选框，然后选择 Save (保存)。要显示有关权限的信息，请选择帮助图标。

c. 公有访问权限

要向一般公众 (世界上的每个人) 授予对您的对象的访问权限，请在 Public access (公开访问) 下面选择 Everyone (所有人)。授予公有访问权限意味着世界上的任何人都可以访问该对象。

选中要授予的权限所对应的复选框，然后选择 Save (保存)。

> **Warning**
>
> - 在授予 Everyone (所有人) 组对您的 Amazon S3 对象的匿名访问权限时应谨慎使用。如果您向此组授予访问权限，那么世界上的任何人都可以访问您的对象。如果您需要为所有人授予访问权限，我们强烈建议您仅授予读取对象的权限。
> - 我们强烈建议您不要为 Everyone (所有人) 组授予写入对象权限。这样做将允许任何人覆盖对象的 ACL 权限。

使用 AWS SDK

本节提供了有关如何在存储桶和对象上配置访问控制列表（ACL）授予的示例。

> **Important**
>
> 如果您的存储桶针对 S3 对象所有权使用强制存储桶拥有者设置，则必须使用策略授予对存储桶及其中对象的访问权限。启用强制存储桶拥有者设置后，设置访问控制列表（ACL）或更新 ACL 的请求将失败并返回 `AccessControlListNotSupported` 错误代码。仍然支持读取 ACL 的请求。

Java

本节提供了有关如何在存储桶和对象上配置访问控制列表（ACL）授予的示例。第一个示例将创建具有标准 ACL（请参阅标准 ACL）的存储桶，创建自定义权限授予列表，然后将标准 ACL 替换为包含自定义授予的 ACL。第二个示例演示如何使用 `AccessControlList.grantPermission()` 方法修改 ACL。

Example 创建存储桶并指定用于向 S3 日志传输组授予权限的标准 ACL

此示例将创建一个存储桶。在请求中，此示例指定了一个标准 ACL，该 ACL 向日志传输组授予将日志写入到存储桶的权限。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.ArrayList;

public class CreateBucketWithACL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String userEmailForReadPermission = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();
```

```
            // Create a bucket with a canned ACL. This ACL will be replaced by the
            // setBucketAcl()
            // calls below. It is included here for demonstration purposes.
            CreateBucketRequest createBucketRequest = new
CreateBucketRequest(bucketName, clientRegion.getName())
                    .withCannedAcl(CannedAccessControlList.LogDeliveryWrite);
            s3Client.createBucket(createBucketRequest);

            // Create a collection of grants to add to the bucket.
            ArrayList<Grant> grantCollection = new ArrayList<Grant>();

            // Grant the account owner full control.
            Grant grant1 = new Grant(new
CanonicalGrantee(s3Client.getS3AccountOwner().getId()),
                    Permission.FullControl);
            grantCollection.add(grant1);

            // Grant the LogDelivery group permission to write to the bucket.
            Grant grant2 = new Grant(GroupGrantee.LogDelivery, Permission.Write);
            grantCollection.add(grant2);

            // Save grants by replacing all current ACL grants with the two we just
created.
            AccessControlList bucketAcl = new AccessControlList();
            bucketAcl.grantAllPermissions(grantCollection.toArray(new Grant[0]));
            s3Client.setBucketAcl(bucketName, bucketAcl);

            // Retrieve the bucket's ACL, add another grant, and then save the new
ACL.
            AccessControlList newBucketAcl = s3Client.getBucketAcl(bucketName);
            Grant grant3 = new Grant(new
EmailAddressGrantee(userEmailForReadPermission), Permission.Read);
            newBucketAcl.grantAllPermissions(grant3);
            s3Client.setBucketAcl(bucketName, newBucketAcl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
```

```
    }
}
```

Example 更新现有对象的 ACL

此示例将更新对象上的 ACL。该示例执行以下任务：

- 检索对象的 ACL
- 通过删除所有现有权限来清除该 ACL
- 添加两个权限：对所有者的完全访问权限以及对通过电子邮件地址标识的用户的 WRITE_ACP 权限 (请参阅 我能授予哪些许可？)。
- 将 ACL 保存到对象

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.CanonicalGrantee;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
import com.amazonaws.services.s3.model.Permission;

import java.io.IOException;

public class ModifyACLExistingObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String emailGrantee = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
```

```
            // Get the existing object ACL that we want to modify.
            AccessControlList acl = s3Client.getObjectAcl(bucketName, keyName);

            // Clear the existing list of grants.
            acl.getGrantsAsList().clear();

            // Grant a sample set of permissions, using the existing ACL owner for
 Full
            // Control permissions.
            acl.grantPermission(new CanonicalGrantee(acl.getOwner().getId()),
 Permission.FullControl);
            acl.grantPermission(new EmailAddressGrantee(emailGrantee),
 Permission.WriteAcp);

            // Save the modified ACL back to the object.
            s3Client.setObjectAcl(bucketName, keyName, acl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

Example 创建存储桶并指定用于向 S3 日志传输组授予权限的标准 ACL

此 C# 示例将创建一个存储桶。在请求中，代码还将指定一个标准 ACL，该 ACL 向日志传输组授予将日志写入到存储桶的权限。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
```

```
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingBucketACLTest
    {
        private const string newBucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateBucketUseCannedACLAsync().Wait();
        }

        private static async Task CreateBucketUseCannedACLAsync()
        {
            try
            {
                // Add bucket (specify canned ACL).
                PutBucketRequest putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = S3Region.EUW1, // S3Region.US,
                                                  // Add canned ACL.
                    CannedACL = S3CannedACL.LogDeliveryWrite
                };
                PutBucketResponse putBucketResponse = await
 client.PutBucketAsync(putBucketRequest);

                // Retrieve bucket ACL.
                GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
                {
                    BucketName = newBucketName
                });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("S3 error occurred. Exception: " +
 amazonS3Exception.ToString());
```

```
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

Example 更新现有对象的 ACL

此 C# 示例将更新现有对象上的 ACL。该示例执行以下任务：

- 检索对象的 ACL。
- 通过删除所有现有权限来清除该 ACL。
- 添加两个权限：对所有者的完全访问权限以及对通过电子邮件地址标识的用户的 WRITE_ACP 权限。
- 通过发送 PutAcl 请求来保存该 ACL。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingObjectACLTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key name ***";
        private const string emailAddress = "*** email address ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
```

```
        {
            client = new AmazonS3Client(bucketRegion);
            TestObjectACLTestAsync().Wait();
        }
        private static async Task TestObjectACLTestAsync()
        {
            try
            {
                    // Retrieve the ACL for the object.
                    GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName
                    });

                    S3AccessControlList acl = aclResponse.AccessControlList;

                    // Retrieve the owner (we use this to re-add permissions after
we clear the ACL).
                    Owner owner = acl.Owner;

                    // Clear existing grants.
                    acl.Grants.Clear();

                    // Add a grant to reset the owner's full permission (the
previous clear statement removed all permissions).
                    S3Grant fullControlGrant = new S3Grant
                    {
                        Grantee = new S3Grantee { CanonicalUser = owner.Id },
                        Permission = S3Permission.FULL_CONTROL

                    };

                    // Describe the grant for the permission using an email address.
                    S3Grant grantUsingEmail = new S3Grant
                    {
                        Grantee = new S3Grantee { EmailAddress = emailAddress },
                        Permission = S3Permission.WRITE_ACP
                    };
                    acl.Grants.AddRange(new List<S3Grant> { fullControlGrant,
grantUsingEmail });

                    // Set a new ACL.
```

```
                    PutACLResponse response = await client.PutACLAsync(new
PutACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        AccessControlList = acl
                    });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

使用 REST API

Amazon S3 API 使您可以在创建存储段或对象时设置 ACL。Amazon S3 还提供 API 以在现有存储段或对象上设置 ACL。这些 API 提供了以下方法来设置 ACL：

- 使用请求标头设置 ACL — 发送创建资源（存储桶或对象）的请求时，您可以使用请求标头设置 ACL。使用这些标头时，您可以指定一个标准 ACL 或者显式指定授权 (显式识别被授权者和许可)。
- 使用请求正文设置 ACL — 当您发送在现有资源上设置 ACL 的请求时，您可以在请求标头或正文中设置 ACL。

有关管理 ACL 的 REST API 支持的信息，请参阅 Amazon Simple Storage Service API 参考中的以下部分：

- GET Bucket acl
- PUT Bucket acl
- GET Object acl
- PUT Object acl
- PUT Object

- PUT Bucket
- PUT Object – 复制
- 开始分段上传

> **Important**
>
> 如果您的存储桶针对 S3 对象所有权使用强制存储桶拥有者设置，则必须使用策略授予对存储桶及其中对象的访问权限。启用强制存储桶拥有者设置后，设置访问控制列表（ACL）或更新 ACL 的请求将失败并返回 `AccessControlListNotSupported` 错误代码。仍然支持读取 ACL 的请求。

访问控制列表 (ACL) 特定的请求标头

您可以使用标头授予基于访问控制列表 (ACL) 的权限。默认情况下，所有的对象都是私有的。只有所有者具有完全访问权限控制。添加新的对象时，您可以向单个 AWS 账户或 Amazon S3 定义的预定义组授予权限。然后，这些权限将添加到对象上的访问控制列表 (ACL)。有关更多信息，请参阅 访问控制列表 (ACL) 概述。

通过此操作，您可以使用以下两种方法之一授予访问权限：

- 标准 ACL (**`x-amz-acl`**) — Amazon S3 支持一系列预定义的 ACL，称为标准 ACL。每个标准 ACL 都有一组预定义的被授权者和许可。有关更多信息，请参阅 标准 ACL。
- 访问权限 – 要向特定 AWS 账户 或组显式授予访问权限，请使用以下标头。每个标头映射到 Amazon S3 在 ACL 中支持的特定权限。有关更多信息，请参阅 访问控制列表 (ACL) 概述。在标头中，您可以指定获得特定权限的被授权者的列表。
  - x-amz-grant-read
  - x-amz-grant-write
  - x-amz-grant-read-acp
  - x-amz-grant-write-acp
  - x-amz-grant-full-control

使用 AWS CLI

有关使用 AWS CLI 管理 ACL 的更多信息，请参阅 AWS CLI 命令参考中的 put-bucket-acl。

> **Important**
>
> 如果您的存储桶针对 S3 对象所有权使用强制存储桶拥有者设置，则必须使用策略授予对存储桶及其中对象的访问权限。启用强制存储桶拥有者设置后，设置访问控制列表（ACL）或更新 ACL 的请求将失败并返回 `AccessControlListNotSupported` 错误代码。仍然支持读取 ACL 的请求。

适用于 ACL 的策略示例

您可在存储桶策略中使用条件键来控制针对 Amazon S3 的访问权限。

主题

- 授予 s3:PutObject 权限，指定了要求存储桶拥有者获得完全控制的条件
- 授予 s3:PutObject 权限，指定了关于 x-amz-acl 标头的条件

授予 s3:PutObject 权限，指定了要求存储桶拥有者获得完全控制的条件

PUT Object 操作允许特定于访问控制列表 (ACL) 的标头，可用于授予基于 ACL 的权限。通过使用这些键，存储桶拥有者可设置条件，要求用户上传对象时需具有特定访问权限。

假设账户 A 拥有一个存储桶，而账户管理员想要授予账户 B 中的用户 Dave 上传对象的权限。默认情况下，Dave 上传的对象由账户 B 拥有，而账户 A 对这些对象没有权限。由于存储桶拥有者要支付账单，需要对 Dave 上传的对象具有全部权限。要实现这一目的，账户 A 管理员可向 Dave 授予 `s3:PutObject` 权限，指定的条件要求请求包含 ACL 特定标头（用于显式授予全部权限）或使用标准 ACL。有关更多信息，请参阅 PUT Object。

需要 x-amz-full-control 标头

您可以要求在请求中具有 `x-amz-full-control` 标头，授予存储桶拥有者完全控制权限。以下存储桶策略向用户 Dave 授予 `s3:PutObject` 权限，使用 `s3:x-amz-grant-full-control` 条件键指定了条件，要求此请求包含 `x-amz-full-control` 标头。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
```

```
      "Principal": {
        "AWS": "arn:aws:iam::AccountB-ID:user/Dave"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::awsexamplebucket1/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
        }
      }
    }
  ]
}
```

> **Note**
>
> 此示例是关于跨账户权限的。不过，如果 Dave（正在获得权限）属于拥有存储桶的 AWS 账户，则该条件权限不是必需的。这是因为，Dave 所属的父账户拥有用户上传的对象。

添加显式拒绝

上述存储桶策略向账户 B 中的用户 Dave 授予条件权限。当该策略生效时，Dave 可通过其他某个策略获得没有任何条件的相同权限。例如，Dave 可能属于一个组，并且您为该组授予 `s3:PutObject` 权限而没有指定任何条件。为避免这些权限漏洞，可通过添加显式拒绝编写更严格的访问策略。在该示例中，如果用户 Dave 的请求没有包含必要标头向存储桶拥有者授予全部权限，则您显式拒绝他的上传权限。显式拒绝始终取代授予的其他任何权限。以下是添加了显式拒绝的经修订的访问策略示例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
```

```
                }
            }
        },
        {
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::AccountB-ID:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        }
    ]
}
```

使用 AWS CLI 测试策略

如果您有两个 AWS 账户，则可使用 AWS Command Line Interface (AWS CLI) 测试此策略。您可以附加此策略，并使用 Dave 的凭证通过以下 AWS CLI `put-object` 命令测试权限。通过添加 `--profile`参数提供 Dave 的凭证。通过添加 `--grant-full-control` 参数可向存储桶拥有者授予完全控制权限。有关设置和使用 AWS CLI 的更多信息，请参阅 使用 AWS CLI 进行 Amazon S3 开发。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --grant-full-control id="AccountA-CanonicalUserID" --profile AccountBUserProfile
```

需要 x-amz-acl 标头

您可以要求 `x-amz-acl` 标头，带有向存储桶拥有者授予完全控制权限的标准 ACL。如果要求在请求中使用 `x-amz-acl` 标头，您可以替换 `Condition` 块中的键值对并指定 `s3:x-amz-acl` 条件键，如以下示例中所示。

```
"Condition": {
    "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control"
    }
}
```

要使用 AWS CLI 测试权限，则指定 `--acl` 参数。然后 AWS CLI 在其发送此请求时添加 `x-amz-acl` 标头。

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --acl "bucket-owner-full-control" --profile AccountBadmin
```

授予 s3:PutObject 权限，指定了关于 x-amz-acl 标头的条件

如果请求包含可使对象公开可读的 `x-amz-acl` 标头，则以下存储桶策略向两个 AWS 账户授予 `s3:PutObject` 权限。`Condition` 块使用 `StringEquals` 条件，并且提供键值对 `"s3:x-amz-acl":["public-read"]`，以进行评估。在该键值对中，`s3:x-amz-acl` 是特定于 Amazon S3 的键，如前缀 `s3:` 所示。

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
         "Sid":"AddCannedAcl",
            "Effect":"Allow",
         "Principal": {
                "AWS": [
                "arn:aws:iam::Account1-ID:root",
    "arn:aws:iam::Account2-ID:root"
    ]
            },
         "Action":"s3:PutObject",
            "Resource": ["arn:aws:s3:::awsexamplebucket1/*"],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl":["public-read"]
                }
            }
        }
    ]
}
```

> **Important**
>
> 并非所有条件对所有操作都有意义。例如，在授予 `s3:CreateBucket` Amazon S3 权限的策略上包含 `s3:LocationConstraint` 条件是有意义的。但是，在授予 `s3:GetObject` 权限的策略中包含此条件没有意义。Amazon S3 可测试是否存在此类涉及 Amazon S3 特定条件的

语义错误。但如果要创建针对 IAM 用户或角色的策略，并且包含了语义上无效的 Amazon S3 条件，则不报告错误，因为 IAM 无法验证 Amazon S3 条件。

阻止对您的 Amazon S3 存储的公有访问

Amazon S3 屏蔽公共访问权限特征提供接入点、存储桶和账户设置，帮助您管理对 Amazon S3 资源的公有访问。默认情况下，新存储桶、接入点和对象不允许公有访问。但是，用户可以修改存储桶策略、接入点策略或对象权限以允许公有访问。S3 屏蔽公共访问权限设置会覆盖这些策略和权限，以便于您可以限制这些资源的公有访问。

借助 S3 屏蔽公共访问权限，账户管理员和存储桶拥有者可以轻松设置集中控制，以控制对已实施的 Amazon S3 资源的公有访问（与资源的创建方式无关）。

有关配置公有块访问的说明，请参阅配置屏蔽公共访问权限。

当 Amazon S3 收到访问存储桶或对象的请求时，它将确定该存储桶或存储桶拥有者的账户是否应用了屏蔽公共访问权限设置。如果请求是通过接入点发出，则 Amazon S3 还会检查接入点的屏蔽公共访问权限设置。如果现有的屏蔽公共访问权限设置禁止请求的访问，则 Amazon S3 将拒绝该请求。

Amazon S3 屏蔽公共访问权限提供四种设置。这些设置彼此独立，可任意组合使用。每个设置都可以应用于接入点、存储桶或整个 AWS 账户。如果接入点、存储桶或账户的屏蔽公共访问权限设置不同，则 Amazon S3 应用接入点、存储桶和账户设置的最严格组合。

当 Amazon S3 评估屏蔽公共访问权限设置是否禁止某一操作时，它将拒绝违反接入点、存储桶或账户设置的任何请求。

Important

通过访问控制列表（ACL）、接入点策略或存储桶策略，或者同时通过这几项向存储桶和对象授予公有访问权限。为了帮助确保您的所有 Amazon S3 接入点、存储桶和对象阻止了公有访问，我们建议为您的账户启用屏蔽公共访问权限的所有四个设置。这些设置阻止所有当前和将来的存储桶和接入点的公有访问。

在应用这些设置之前，请确认您的应用程序在没有公有访问的情况下能够正常工作。如果您需要对存储桶或对象进行某种级别的公有访问，例如，按照 使用 Amazon S3 托管静态网站 中所述托管静态网站，则可以自定义各个设置以符合您的存储使用案例要求。

启用“屏蔽公共访问权限”可防止通过直接附加到 S3 资源的资源策略或访问控制列表（ACL）来授予公共访问权限，有助于保护您的资源。除了启用“屏蔽公共访问权限”之外，还要仔细检查以下策略，来确认它们不会授予公共访问权限：

- 附加到关联 AWS 主体（例如 IAM 角色）的基于身份的策略
- 附加到关联 AWS 资源 [例如 AWS Key Management Service（KMS）密钥] 的基于资源的策略

> **Note**
>
> - 您只能为接入点、存储桶和 AWS 账户启用屏蔽公共访问权限设置。Amazon S3 不支持基于每个对象的屏蔽公共访问权限设置。
> - 在将屏蔽公共访问权限设置应用于某一账户时，这些设置将应用于全球所有 AWS 区域。这些设置可能不会立即或同时在所有区域生效，但最终会传播到所有区域。

主题

屏蔽公共访问权限设置

S3 屏蔽公共访问权限提供四种设置。您可以任意组合将这些设置应用于单个接入点、存储桶或整个 AWS 账户。如果您将某一设置应用于某个账户，则该设置将应用于该账户拥有的所有存储桶和接入点。同样，如果您将设置应用于某个存储桶，则该设置将应用于与该存储桶关联的所有接入点。

下表包含可用设置。

名称	描述
`BlockPublicAcls`	将此选项设置为 TRUE 会引发以下行为： • 如果指定的访问控制列表（ACL）为公有，PUT Bucket acl 和 PUT Object acl 将失败。 • 如果请求包含公有 ACL，则 PUT Object 调用失败。 • 如果将此设置应用于某个账户，则当请求包含公有 ACL 时，PUT Bucket 调用将失败。 将此设置设为 TRUE 时，指定操作将失败（通过 REST API、AWS CLI 或 AWS SDK 请求的操作）。但是，不修改用于存储桶和对象的现有策略和 ACL。此设置不仅允许您针对公有访问提供保护，还允许您审核、优化或更改存储桶或对象的现有策略和 ACL。 **Note** 接入点没有与其关联的 ACL。如果您将此设置应用于接入点，则该设置将充当到底层存储桶的传递途径。如果某个接入点启用了此设置，则无论存储桶实际上是否启用了此设置，通过该接入点发出的请求表现为就像底层存储桶启用了此设置。
`IgnorePublicAcls`	将此选项设置为 TRUE 会使 Amazon S3 忽略存储桶及其包含的任何对象上的所有公有 ACL。通过此设置，您可以安全地阻止 ACL 授权的公有访问，同时仍允许包含公有 ACL 的 PUT Object 调用（与 `BlockPublicAcls` 相反，后者拒绝包含公有 ACL 的 PUT Object 调用）。启用此设置不影响任何现有 ACL 的持久性，也不会阻止设置新的公有 ACL。 **Note** 接入点没有与其关联的 ACL。如果您将此设置应用于接入点，则该设置将充当到底层存储桶的传递途径。如果某个接入点启用了此设置，

名称	描述
	则无论存储桶实际上是否启用了此设置，通过该接入点发出的请求表现为就像底层存储桶启用了此设置。
BlockPublicPolicy	对于存储桶将此选项设置为 TRUE 将导致 Amazon S3 在指定的存储桶策略允许公有访问时拒绝对 PUT 存储桶策略的调用。如果指定的策略允许公有访问，则对于存储桶将此选项设置为 TRUE 也会导致 Amazon S3 拒绝对存储桶的所有相同账户接入点的 PUT 接入点策略的调用。 如果（为接入点或底层存储桶）指定的策略允许公有访问，则对于接入点将此选项设置为 TRUE 会导致 Amazon S3 拒绝通过该接入点对 PUT 接入点策略和 PUT 存储桶策略的调用。 您可以使用此设置以允许用户管理接入点和存储桶策略，而不允许其公开共享存储桶或其包含的对象。启用此设置不会影响现有接入点或存储桶策略。 **Important** 要有效地使用此设置，我们建议您在账户级别应用此设置。存储桶策略可能允许用户更改存储桶的屏蔽公共访问权限设置。因此，有权更改存储桶策略的用户可以插入允许其为存储桶禁用屏蔽公共访问权限设置的策略。如果为整个账户而非特定存储桶启用此设置，即使用户更改存储桶策略以禁用此设置，Amazon S3 仍会阻止公有策略。
RestrictPublicBuckets	将此选项设置为 TRUE 会将对具有公有策略的接入点或存储桶的访问限制为该存储桶拥有者账户和接入点拥有者账户中的 AWS 服务主体和授权用户。此设置会阻止对接入点或存储桶的所有跨账户访问（AWS 服务主体的访问除外），但仍允许该账户内的用户管理接入点或存储桶。 启用此设置不影响现有的接入点或存储桶策略，但 Amazon S3 会阻止派生自任何公有接入点或存储桶策略的公有和跨账户访问，包括到特定账户的非公有委派。

Important

- 调用 GET Bucket acl 和 GET Object acl 始终返回指定存储桶或对象的已就位的有效权限。例如，假设存储桶有一个 ACL 用于授予公有访问权限，但该存储桶还启用了 `IgnorePublicAcls` 设置。在此情况下，GET Bucket acl 将返回可反映 Amazon S3 正在实施的访问权限的 ACL，而不是与存储桶关联的实际 ACL。
- 屏蔽公共访问权限设置不更改现有的策略或 ACL。因此，删除屏蔽公共访问权限设置会使具有公有策略或 ACL 的存储桶或对象再次变为可公开访问。

在接入点上执行屏蔽公共访问权限操作

为了对接入点执行屏蔽公共访问权限操作，请使用 AWS CLI 服务 `s3control`。

Important

请注意，目前，在创建接入点后无法再更改接入点的屏蔽公共访问权限设置。因此，指定接入点的屏蔽公共访问权限设置的唯一方法是在创建接入点时包含这些设置。

“公有”的含义

ACL

如果存储桶或对象 ACL 可向预定义的 `AllUsers` 或 `AuthenticatedUsers` 组的成员授予任何权限，则 Amazon S3 将其视为公有。有关预定义组的更多信息，请参阅 Amazon S3 预定义的组。

存储桶策略

在评估存储桶策略时，Amazon S3 先假定该策略是公有的。然后对策略进行评估，以确定它是否符合非公有条件。当存储桶策略必须仅授予针对以下一个或多个项目的固定值（不包含通配符或 AWS Identity and Access Management 策略变量的值）的访问权限时，才会将该策略视为非公有：

- AWS 主体、用户、角色或服务主体（例如 `aws:PrincipalOrgID`）
- 一组无类域间路由 (CIDR)，使用 `aws:SourceIp`。有关 CIDR 的更多信息，请参阅 RFC 编辑器网站上的 RFC 4632。

> **Note**
>
> 根据 aws:SourceIp 条件键授予访问权限且 IP 范围非常宽泛（例如 0.0.0.0/1）的存储桶策略将评估为“公有”。这包括大于 /8（对于 IPv4）和 /32（对于 IPv6，不包括 RFC1918 私有范围）的值。屏蔽公共访问权限将拒绝这些“公有”策略，并阻止跨账户访问已在使用这些“公有”策略的存储桶。

- aws:SourceArn
- aws:SourceVpc
- aws:SourceVpce
- aws:SourceOwner
- aws:SourceAccount
- aws:userid，在模式“AROLEID:*”之外
- s3:DataAccessPointArn

> **Note**
>
> 在存储桶策略中使用时，只要账户 ID 已固定，此值就可以包含接入点名称的通配符，而无需将策略呈现为公有。例如，允许访问 arn:aws:s3:us-west-2:123456789012:accesspoint/* 也会允许访问与区域 123456789012 中的账户 us-west-2 关联的任何接入点，而无需将存储桶策略呈现为公有。请注意，对于接入点策略，此行为会有不同。有关更多信息，请参阅 接入点。

- s3:DataAccessPointAccount

有关存储桶策略的更多信息，请参阅 Amazon S3 的存储桶策略。

Example ：公有存储桶策略

根据这些规则，以下示例策略被视为公有。

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow"
```

```
 }
```

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": { "StringLike": {"aws:SourceVpc": "vpc-*"}}
 }
```

您可以通过使用固定值来包含上面所列的任何条件键将这些策略变为非公有。例如，可以通过将 aws:SourceVpc 设置为固定值，将上面的最后一个策略变为非公有，类似于以下内容：

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": {"StringEquals": {"aws:SourceVpc": "vpc-91237329"}}
 }
```

Amazon S3 如何评估同时包含公有和非公有访问授权的存储桶策略

此示例演示 Amazon S3 如何评估同时包含公有和非公有访问授权的存储桶策略。

假设存储桶有一个策略可向一组固定主体授予访问权限。根据前面描述的规则，此策略不是公有策略。因此，如果您启用 RestrictPublicBuckets 设置，该策略将按编写内容保持有效，因为 RestrictPublicBuckets 只应用于具有公有策略的存储桶。但是，如果您向该策略中添加公有语句，则 RestrictPublicBuckets 将对该存储桶生效。它仅允许存储桶拥有者账户的 AWS 服务主体和授权用户访问该存储桶。

例如，假定“Account-1”拥有的一个存储桶包含以下内容：

1. 向 AWS CloudTrail（它是 AWS 服务主体）授予访问权限的语句。
2. 向账户“Account-2”授予访问权限的语句
3. 向公众授予访问权限的语句，例如通过指定 "Principal": "*" 且没有限制性的 Condition

由于第三条语句，此策略符合公有策略的条件。实施此策略并启用 RestrictPublicBuckets 后，Amazon S3 将仅允许 CloudTrail 进行访问。请注意，尽管语句 2 不是公有，但

Amazon S3 仍禁用“Account-2”的访问。这是因为，语句 3 将整个策略都渲染为公有，因此 `RestrictPublicBuckets` 适用。因此，即使策略向特定账户“Account-2”委派访问权限，Amazon S3 仍禁用跨账户访问。但如果您从该策略中删除语句 3，则该策略不符合公有条件，并且 `RestrictPublicBuckets` 不再适用。因此，即使将 `RestrictPublicBuckets` 保持启用状态，“Account-2”也会重获存储桶的访问权限。

接入点

与存储桶相比，Amazon S3 评估接入点的屏蔽公共访问权限设置略有不同。Amazon S3 用于确定接入点策略何时为公用的规则对接入点和存储桶来说通常是相同的，但以下情况除外：

- 具有 VPC 网络起源的接入点始终被视为非公有，而无论其接入点策略的内容如何。
- 使用 `s3:DataAccessPointArn` 向一组接入点授予访问权限的接入点策略被视为公有。请注意，此行为与存储桶策略不同。例如，对于与 `s3:DataAccessPointArn` 匹配的 `arn:aws:s3:us-west-2:123456789012:accesspoint/*` 值授予访问权限的存储桶策略不会被视为公有。但是，在接入点策略中使用相同的语句会使接入点变为公有。

使用适用于 S3 的 IAM Access Analyzer 查看公有存储桶

您可以使用适用于 S3 的 IAM Access Analyzer 来查看具有存储桶 ACL、存储桶策略或授予公共访问权限的接入点策略的存储桶。如果存在已配置为允许互联网上的任何人或其他 AWS 账户（包括组织外部的 AWS 账户）访问的存储桶，适用于 S3 的 IAM Access Analyzer 会向您发出提醒。您会收到每个公共存储桶或共享存储桶的结果，其中报告了公共或共享访问的来源和级别。

在适用于 S3 的 IAM Access Analyzer 中，只需单击一下即可屏蔽对存储桶的所有公共访问权限。您还可以向下钻取到存储桶级别权限设置，以配置精细访问。对于需要公共或共享访问的特定和经验证的使用案例，您可以通过对存储桶的结果进行归档来确认和记录存储桶保持公开或共享的意图。

在极少数情况下，对于 Amazon S3 屏蔽公共访问权限评估报告为公有的存储桶，适用于 S3 的 IAM Access Analyzer 可能不会报告任何调查发现。发生这种情况的原因是，Amazon S3 屏蔽公共访问权限会审核当前操作以及将来可能添加的任何潜在操作的策略，从而导致存储桶变为公有。另一方面，适用于 S3 的 IAM Access Analyzer 只分析在评估访问状态时为 Amazon S3 服务指定的当前操作。

有关适用于 S3 的 IAM Access Analyzer 的更多信息，请参阅[使用适用于 S3 的 IAM Access Analyzer 查看存储桶访问权限]。

权限

要使用 Amazon S3 屏蔽公共访问权限特征，您必须拥有以下权限。

操作	所需权限
GET 存储桶策略状态	`s3:GetBucketPolicyStatus`
GET 存储桶屏蔽公共访问权限设置	`s3:GetBucketPublicAccessBlock`
PUT 存储桶屏蔽公共访问权限设置	`s3:PutBucketPublicAccessBlock`
DELETE 存储桶屏蔽公共访问权限设置	`s3:PutBucketPublicAccessBlock`
GET 账户屏蔽公共访问权限设置	`s3:GetAccountPublicAccessBlock`
PUT 账户屏蔽公共访问权限设置	`s3:PutAccountPublicAccessBlock`
DELETE 账户屏蔽公共访问权限设置	`s3:PutAccountPublicAccessBlock`
PUT 接入点屏蔽公共访问权限设置	`s3:CreateAccessPoint`

> **Note**
>
> DELETE 操作所需权限与 PUT 操作相同。没有单独用于 DELETE 操作的权限。

配置屏蔽公共访问权限

有关为 AWS 账户、Amazon S3 存储桶和接入点配置屏蔽公共访问权限的更多信息，请参阅以下主题：

- 为您的账户配置屏蔽公共访问权限设置
- 为 S3 存储桶配置屏蔽公共访问权限设置
- 在接入点上执行屏蔽公共访问权限操作

为您的账户配置屏蔽公共访问权限设置

Amazon S3 屏蔽公共访问权限提供接入点、存储桶和账户的设置，帮助您管理对 Amazon S3 资源的公有访问。默认情况下，新存储桶、接入点和对象不允许公有访问。

有关更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

> **Note**
>
> 账户级别设置会覆盖各个对象的设置。将您的账户配置为屏蔽公共访问权限后，将覆盖对账户内单个对象所设定的任何公共访问权限设置。

您可以使用 S3 控制台、AWS CLI、AWS SDK 和 REST API 为账户中的所有存储桶配置屏蔽公共访问权限设置。有关更多信息，请参阅以下部分。

要为存储桶配置屏蔽公共访问权限设置，请参阅 为 S3 存储桶配置屏蔽公共访问权限设置。有关接入点的信息，请参阅 在接入点上执行屏蔽公共访问权限操作。

使用 S3 控制台

Amazon S3 屏蔽公共访问权限将阻止所有在设置中允许对 S3 存储桶中的数据进行公有访问的应用程序。本部分介绍如何为 AWS 账户 中的所有 S3 存储桶编辑屏蔽公共访问权限设置。有关屏蔽公共访问权限的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

为 AWS 账户 中的所有 S3 存储桶编辑屏蔽公共访问权限设置

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 请选择 Block Public Access settings for this account (阻止此账户的公有访问设置)。
3. 请选择 Edit（编辑）以便为 AWS 账户 中的所有存储桶更改屏蔽公共访问权限设置。
4. 请选择要更改的设置，然后选择 Save changes (保存更改)。
5. 当系统要求确认时，请输入 **`confirm`**。然后选择 Confirm (确认) 以保存更改。

使用 AWS CLI

您可以通过 AWS CLI 使用 Amazon S3 屏蔽公共访问权限。有关设置和使用 AWS CLI 的更多信息，请参阅什么是 AWS Command Line Interface？

账户

为了对账户执行屏蔽公共访问权限操作，请使用 AWS CLI 服务 `s3control`。使用此服务的账户级别的操作如下所示：

- PUT PublicAccessBlock（用于账户）
- GET PublicAccessBlock（用于账户）

- DELETE PublicAccessBlock（用于账户）

有关其他信息和示例，请参阅 AWS CLI 参考中的 put-public-access-block。

使用 AWS SDK

Java

以下示例展示了如何将 Amazon S3 屏蔽公共访问权限与 AWS SDK for Java 结合使用，以在 Amazon S3 账户中放置公有访问阻止配置。

```
AWSS3ControlClientBuilder controlClientBuilder =
 AWSS3ControlClientBuilder.standard();
controlClientBuilder.setRegion(<region>);
controlClientBuilder.setCredentials(<credentials>);

AWSS3Control client = controlClientBuilder.build();
client.putPublicAccessBlock(new PutPublicAccessBlockRequest()
  .withAccountId(<account-id>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withIgnorePublicAcls(<value>)
    .withBlockPublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

> **Important**
>
> 此示例仅适用于使用 `AWSS3Control` 客户端类的账户级别操作。对于存储桶级别的操作，请参阅前面的示例。

Other SDKs

有关使用其他 AWS SDK 的信息，请参阅 使用 AWS SDK 通过 Amazon S3 进行开发。

使用 REST API

有关通过 REST API 使用 Amazon S3 屏蔽公共访问权限的信息，请参阅 Amazon Simple Storage Service API 参考中的以下主题。

- 账户级别操作
  - PUT PublicAccessBlock
  - GET PublicAccessBlock
  - DELETE PublicAccessBlock

为 S3 存储桶配置屏蔽公共访问权限设置

Amazon S3 屏蔽公共访问权限提供接入点、存储桶和账户的设置，帮助您管理对 Amazon S3 资源的公有访问。默认情况下，新存储桶、接入点和对象不允许公有访问。

有关更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

您可以使用 S3 控制台、AWS CLI、AWS SDK 和 REST API 授予对一个或多个存储桶的公共访问权限。您还可以对已经设为公共的存储桶屏蔽公共访问权限。有关更多信息，请参阅以下部分。

要为账户中的每个存储桶配置“屏蔽公共访问权限”设置，请参阅为您的账户配置屏蔽公共访问权限设置。有关为接入点配置屏蔽公共访问权限的信息，请参阅 在接入点上执行屏蔽公共访问权限操作。

使用 S3 控制台

Amazon S3 屏蔽公共访问权限将阻止使用允许对 S3 存储桶中的数据进行公有访问的任何设置的应用程序。本部分介绍如何为一个或多个 S3 存储桶编辑屏蔽公共访问权限设置。有关使用 AWS CLI、AWS SDK 和 Amazon S3 REST API 阻止公有访问的信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

您可以在存储桶列表中查看您的存储桶是否可以公开访问。在访问列中，Amazon S3 会标注存储桶的权限，如下所示：

- 公有 – 所有人都拥有以下一项或多项访问权限：列出对象、写入对象、读取和写入权限。
- 对象可以是公有的 – 存储桶不是公有的，但具有适当权限的任何人都可以授予对象公有访问权限。
- 存储桶和对象不是公有的 – 存储桶和对象没有任何公有访问权限。
- 仅限此账户的授权用户 – 由于存在授予公有访问权限的策略，因此访问权限仅限于此账户中的 IAM 用户和角色以及 AWS 服务主体。

您还可以按访问类型来筛选存储桶搜索。从 Search for bucket（搜索存储桶）栏旁边的下拉列表中选择一种访问类型。

如果您在列出存储桶及其公有访问设置时看到 `Error`，则您可能不具备所需的权限。检查以确保您已将以下权限添加到您的用户或角色策略：

```
s3:GetAccountPublicAccessBlock
s3:GetBucketPublicAccessBlock
s3:GetBucketPolicyStatus
s3:GetBucketLocation
s3:GetBucketAcl
s3:ListAccessPoints
s3:ListAllMyBuckets
```

在极少数情况下，请求也可能因 AWS 区域中断而失败。

为单个 S3 存储桶编辑 Amazon S3 屏蔽公共访问权限设置

如何您需要为单个 S3 存储桶更改公有访问设置，请执行以下步骤。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket name（存储桶名称）列表中，请选择所需的存储桶的名称。
3. 选择 Permissions (权限)。
4. 请选择 Edit（编辑）以更改存储桶的公有访问设置。有关四个 Amazon S3 屏蔽公共访问权限设置的更多信息，请参阅 屏蔽公共访问权限设置。
5. 请选择要更改的设置，然后选择 Save (保存)。
6. 当系统要求确认时，请输入 **`confirm`**。然后选择 Confirm (确认) 以保存更改。

在创建存储桶时，可以更改 Amazon S3 屏蔽公共访问权限设置。有关更多信息，请参阅 创建存储桶。

使用 AWS CLI

要对存储桶屏蔽公共访问权限或删除公共访问权限屏蔽，请使用 AWS CLI 服务 `s3api`。使用此服务的存储桶级别的操作如下所示：

- PUT PublicAccessBlock（用于存储桶）
- GET PublicAccessBlock（用于存储桶）
- DELETE PublicAccessBlock（用于存储桶）
- GET BucketPolicyStatus

有关更多信息和示例，请参阅 AWS CLI 参考中的 put-public-access-block。

使用 AWS SDK

Java

```
AmazonS3 client = AmazonS3ClientBuilder.standard()
   .withCredentials(<credentials>)
   .build();

client.setPublicAccessBlock(new SetPublicAccessBlockRequest()
  .withBucketName(<bucket-name>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withBlockPublicAcls(<value>)
    .withIgnorePublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

> **Important**
>
> 此示例仅适用于使用 AmazonS3 客户端类的存储桶级别操作。对于账户级别的操作，请参阅以下示例。

Other SDKs

有关使用其他 AWS SDK 的信息，请参阅 使用 AWS SDK 通过 Amazon S3 进行开发。

使用 REST API

有关通过 REST API 使用 Amazon S3 屏蔽公共访问权限的信息，请参阅 Amazon Simple Storage Service API 参考中的以下主题。

- 存储桶级别操作
  - PUT PublicAccessBlock
  - GET PublicAccessBlock
  - DELETE PublicAccessBlock
  - GET BucketPolicyStatus

使用适用于 S3 的 IAM Access Analyzer 查看存储桶访问权限

如果存在已配置为允许互联网上的任何人或其他 AWS 账户（包括组织外部的 AWS 账户）访问的 S3 存储桶，适用于 S3 的 IAM Access Analyzer 会向您发出提醒。对于每个公共存储桶或共享存储桶，您会收到有关公共或共享访问的来源和级别的信息。例如，适用于 S3 的 IAM Access Analyzer 可能会显示存储桶具有通过存储桶访问控制列表（ACL）、存储桶策略，多区域接入点策略或接入点策略提供的读取或写入访问权限。有了这些调查结果，您就可以立即采取精确的纠正措施，将存储桶访问权限恢复为您期望的设置。

在适用于 S3 的 IAM Access Analyzer 中查看存在风险的存储桶时，只需单击一下即可对存储桶屏蔽所有公共访问权限。我们建议您阻止所有对存储桶的访问，除非您需要公有访问才能支持特定使用案例。在阻止所有公有访问之前，请确保您的应用程序在没有公有访问权限的情况下可以继续正常工作。有关更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

您还可以向下钻取到存储桶级别权限设置，以配置精细访问。对于需要公有访问的特定和经验证的使用案例（例如静态网站托管、公共下载或跨账户共享），您可以通过对存储桶的结果进行归档来确认和记录存储桶保持公开或共享的意图。您可以随时重新访问和修改这些存储桶配置。您还可以将结果下载为 CSV 格式的报告以供审计使用。

Amazon S3 控制台上提供适用于 S3 的 IAM Access Analyzer，无需额外费用。适用于 S3 的 IAM Access Analyzer 由 AWS Identity and Access Management（IAM）IAM Access Analyzer 提供支持。要在 Amazon S3 控制台中使用适用于 S3 的 IAM Access Analyzer，您必须访问 IAM 控制台，然后按区域启用 IAM Access Analyzer。

有关 IAM Access Analyzer 的更多信息，请参阅《IAM 用户指南》中的什么是 IAM Access Analyzer？有关适用于 S3 的 IAM Access Analyzer 的更多信息，请查看以下部分。

Important

- 适用于 S3 的 IAM Access Analyzer 需要账户级分析器。要使用适用于 S3 的 IAM Access Analyzer，您必须访问 IAM Access Analyzer 并创建具有一个账户作为信任区域的分析器。有关更多信息，请参阅《IAM 用户指南》中的启用 IAM Access Analyzer。
- 适用于 S3 的 IAM Access Analyzer 不分析附加到跨账户接入点的接入点策略。之所以出现这种行为，是因为接入点及其策略位于信任区域（即账户）之外。如果您尚未将 `RestrictPublicBuckets` 屏蔽公共访问权限设置应用于存储桶或账户，则在 Buckets with public access（具有公有访问的存储桶）下列出了委托访问跨账户接入点的存储桶。当您应用 `RestrictPublicBuckets` 屏蔽公共访问权限设置时，该存储桶将在可从其他 AWS 账户（包括第三方 AWS 账户）访问的存储桶下报告。

- 添加或修改存储桶策略或存储桶 ACL 后，IAM Access Analyzer 会在 30 分钟内根据更改生成和更新调查发现。与账户级屏蔽公共访问权限设置相关的调查结果可能在您更改设置后长达 6 小时无法生成或更新。创建、删除多区域接入点或更改其策略后，最多可能会在 6 小时内生成或更新与多区域接入点相关的调查结果。

主题

适用于 S3 的 IAM Access Analyzer 提供哪些信息？

适用于 S3 的 IAM Access Analyzer 提供可在 AWS 账户之外访问的存储桶的调查发现。Internet 上的任何人均可访问在 Buckets with public access（具有公有访问权限的存储桶）下面列出的存储桶。如果适用于 S3 的 IAM Access Analyzer 识别出公有存储桶，您还会在页面顶部看到一条警告，其中显示您的区域中公有存储桶的数量。可从其他 AWS 账户（包括第三方 AWS 账户）访问的存储桶下面列出的存储桶将与其他 AWS 账户（包括组织外部的账户）有条件地共享。

对于每个存储桶，适用于 S3 的 IAM Access Analyzer 提供以下信息：

- Bucket name（存储桶名称）
- 被 Access Analyzer 发现 - 当适用于 S3 的 IAM Access Analyzer 发现了公有存储桶或共享存储桶访问时。
- 共享方式 - 如何通过存储桶策略、存储桶 ACL 或多区域接入点策略或接入点策略共享存储桶。多区域接入点和跨账户接入点反映在接入点下。可以通过策略和 ACL 来共享存储桶。如果您想要找到并查看存储桶访问的来源，您可以使用此列中的信息作为起点，立即采取精确的纠正措施。
- Status（状态）- 存储桶结果的状态。适用于 S3 的 IAM Access Analyzer 显示所有公有存储桶和共享存储桶的调查发现。

  - Active (活动) - 结果尚未审核。
  - Archived (已归档) - 结果已按预期审核和确认。
  - All（全部）– 公共存储桶或与其他 AWS 账户（包括组织外部的 AWS 账户）共享的存储桶的所有结果。
- Access level（访问级别）- 为存储桶授予的访问权限：
  - List (列出) - 列出资源。
  - Read (读取) - 读取但不编辑资源内容和属性。
  - Write (写入) - 创建、删除或修改资源。
  - Permissions (权限) - 授予或修改资源权限。
  - Tagging (标记) - 更新与资源关联的标记。

启用适用于 S3 的 IAM Access Analyzer

要使用适用于 S3 的 IAM Access Analyzer，您必须完成以下先决条件步骤。

1. 授予所需的权限。

   有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 所需的权限。
2. 请访问 IAM 以便为要使用 IAM Access Analyzer 的每个区域创建账户级分析器。

   适用于 S3 的 IAM Access Analyzer 需要账户级分析器。要使用适用于 S3 的 IAM Access Analyzer，您必须创建具有一个账户作为信任区域的分析器。有关更多信息，请参阅《IAM 用户指南》中的启用 IAM Access Analyzer。

阻止所有公有访问

如果您想要通过单击一次对存储桶屏蔽所有访问权限，则可以使用适用于 S3 的 IAM Access Analyzer 中的屏蔽所有公共访问权限按钮。当您阻止对存储桶的所有公有访问时，系统不会授予公有访问权限。我们建议您阻止所有对存储桶的公有访问，除非您需要公有访问才能支持特定和经验证的使用案例。在阻止所有公有访问之前，请确保您的应用程序在没有公有访问权限的情况下可以继续正常工作。

如果您不想阻止对存储桶的所有公有访问，则可以在 Amazon S3 控制台上编辑屏蔽公共访问权限设置，以配置存储桶的精细访问级别。有关更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

在极少数情况下，对于 Amazon S3 屏蔽公共访问权限评估报告为公有的存储桶，适用于 S3 的 IAM Access Analyzer 可能不会报告任何调查发现。发生这种情况的原因是，Amazon S3 屏蔽公共访问权

限会审核当前操作以及将来可能添加的任何潜在操作的策略，从而导致存储桶变为公有。另一方面，适用于 S3 的 IAM Access Analyzer 只分析在评估访问状态时为 Amazon S3 服务指定的当前操作。

使用适用于 S3 的 IAM Access Analyzer 对存储桶屏蔽所有公共访问权限

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中的 Dashboards（控制面板）下，请选择 Access analyzer for S3（S3 的访问分析器）。
3. 在适用于 S3 的 IAM Access Analyzer 中，选择一个存储桶。
4. 请选择 Block all public access (阻止所有公有访问)。
5. 要确认您阻止对存储桶的所有公有访问的意图，请在 Block all public access（bucket settings）[阻止所有公有访问（存储桶设置）] 中输入 **`confirm`**。

   Amazon S3 会阻止对存储桶的所有公有访问。存储桶调查发现的状态更新为已解决，并且存储桶从适用于 S3 的 IAM Access Analyzer 列表中消失。如果要查看已解决的桶，请在 IAM 控制台上打开 IAM Access Analyzer。

查看和更改桶访问权限

如果不打算对公有账户或其他 AWS 账户（包括组织外部的账户）授予访问权限，可以修改存储桶 ACL、存储桶策略，多区域接入点策略或接入点策略来删除对存储桶的访问权限。Shared through（共享方式）列显示存储桶访问权限的所有来源：存储桶策略、存储桶 ACL 和/或接入点策略。多区域接入点和跨账户接入点反映在接入点下。

查看和更改存储桶策略、存储桶 ACL，多区域接入点策略或接入点策略

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，请选择 Access analyzer for S3 (S3 访问分析器)。
3. 要查看是否已通过存储桶策略、存储桶 ACL 多区域接入点策略或接入点策略授予公有访问权限或共享访问权限，请查看共享方式列。
4. 在存储桶下，请选择您想要更改或查看其存储桶策略、存储桶 ACL 多区域接入点策略或接入点策略的存储桶的名称。
5. 如果要更改或查看存储桶 ACL：

   a. 选择权限。

b. 请选择访问控制列表。

c. 查看您的存储桶 ACL，并根据需要进行更改。

有关更多信息，请参阅 配置 ACL。

6. 如果要更改或查看存储桶策略：

a. 选择权限。

b. 请选择存储桶策略。

c. 根据需要查看或更改存储桶策略。

有关更多信息，请参阅 使用 Amazon S3 控制台添加存储桶策略。

7. 如果要更改或查看多区域接入点策略：

a. 请选择多区域接入点。

b. 请选择多区域接入点名称。

c. 根据需要查看或更改多区域接入点策略。

有关更多信息，请参阅 权限。

8. 如果要查看或更改接入点策略：

a. 请选择接入点。

b. 请选择接入点名称。

c. 根据需要查看或更改访问权限。

有关更多信息，请参阅 通过 Amazon S3 控制台使用 Amazon S3 接入点。

如果您编辑或删除存储桶 ACL、存储桶策略或接入点策略以删除公有访问或共享访问权限，则存储桶结果的状态会更新为已解决。已解决的桶调查发现从适用于 S3 的 IAM Access Analyzer 列表中消失，但您可以在 IAM Access Analyzer 中查看它们。

对存储桶结果进行归档

如果存储桶对公共账户或其他 AWS 账户（包括组织外部的账户）授予访问权限，以便支持特定使用案例（例如，静态网站、公共下载或跨账户共享），则可以归档该存储桶的结果。当您对存储桶结果进行归档时，即表示您确认并记录存储桶保持公开或共享的意图。已归档的存储桶调查发现将保留在适用于 S3 的 IAM Access Analyzer 列表中，以便您始终知道哪些存储桶是公有存储桶或共享存储桶。

在适用于 S3 的 IAM Access Analyzer 中归档存储桶调查发现

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，请选择 Access analyzer for S3（S3 访问分析器）。
3. 在适用于 S3 的 IAM Access Analyzer 中，选择一个活动的存储桶。
4. 要确认您想让公共账户或其他 AWS 账户（包括组织外部的账户）访问此存储桶的意图，请选择 Archive（归档）。
5. 输入 **`confirm`**，然后选择 Archive (归档)。

激活已归档的存储桶结果

对结果进行归档后，您可以随时重新访问这些结果并将其状态更改回活动状态，这表明存储桶需要另一次审核。

在适用于 S3 的 IAM Access Analyzer 中激活归档的存储桶调查发现

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，请选择 Access analyzer for S3（S3 访问分析器）。
3. 请选择已归档的存储桶结果。
4. 请选择 Mark as active (标记为活动)。

查看结果详细信息

如果您需要查看有关桶的更多信息，可以在 IAM 控制台上的 IAM Access Analyzer 中打开桶调查发现详细信息。

在适用于 S3 的 IAM Access Analyzer 中查看调查发现详细信息

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，请选择 Access analyzer for S3（S3 访问分析器）。
3. 在适用于 S3 的 IAM Access Analyzer 中，选择一个存储桶。
4. 请选择 View details (查看详细信息)。

   调查发现详细信息将在 IAM 控制台上的 IAM Access Analyzer 中打开。

下载适用于 S3 的 IAM Access Analyzer 报告

您可以将存储桶结果下载为 CSV 格式的报告，供审计使用。该报告包含的信息与您在 Amazon S3 控制台的适用于 S3 的 IAM Access Analyzer 中看到的信息相同。

下载报告

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，请选择 Access analyzer for S3（S3 的访问分析器）。
3. 在“Region (区域)”筛选器中，请选择相应的区域。

   适用于 S3 的 IAM Access Analyzer 会更新以显示所选区域的存储桶。
4. 请选择 Download report (下载报告)。

   系统会生成 CSV 报告并保存到您的计算机中。

使用存储桶拥有者条件验证存储桶所有权

Amazon S3 存储桶拥有者条件可确保您在 S3 操作中使用的存储桶属于您期望的 AWS 账户。

大多数 S3 操作从特定 S3 存储桶读取或写入特定 S3 存储桶。这些操作包括上传、复制和下载对象、检索或修改存储桶配置以及检索或修改对象配置。执行这些操作时，您可以通过在请求中包含存储桶名称来指定要使用的存储桶。例如，要从 S3 检索对象，您需要发出一个请求，指定存储桶的名称和要从该存储桶中检索的对象键。

由于 Amazon S3 会根据存储桶的名称识别存储桶，因此在请求中使用错误存储桶名称的应用程序可能会无意中对非预期存储桶执行操作。为了帮助避免此类情况下的意外存储桶交互，您可以使用存储桶拥有者条件。存储桶拥有者条件使您能够验证目标存储桶是否由预期 AWS 账户拥有，从而为 S3 操作具有预期效果提供了额外保证。

主题

何时使用存储桶拥有者条件

我们建议您在执行受支持的 S3 操作并知道预期存储桶拥有者的账户 ID 时使用存储桶拥有者条件。存储桶拥有者条件适用于所有 S3 对象操作和大多数 S3 存储桶操作。有关不支持存储桶拥有者条件的 S3 操作的列表，请参阅 限制和局限性。

要了解使用存储桶拥有者条件的好处，请考虑以下涉及 AWS 客户 Bea 的场景：

1. Bea 开发使用 Amazon S3 的应用程序。在开发过程中，Bea 使用仅限测试的 AWS 账户创建名为 `bea-data-test` 的存储桶，并将其应用程序配置为向 `bea-data-test` 发出请求。
2. Bea 部署其应用程序，但忘记将应用程序重新配置为使用其生产 AWS 账户中的存储桶。
3. 在生产中，Bea 的应用程序向 `bea-data-test` 发出请求，这成功了。这导致生产数据写入 Bea 测试账户中的存储桶。

Bea 可以通过使用存储桶拥有者条件来帮助防范此类情况。使用存储桶拥有者条件，Bea 可以在其请求中包含预期存储桶拥有者的 AWS 账户 ID。然后，Amazon S3 在处理每个请求之前检查存储桶拥有者的账户 ID。如果实际存储桶拥有者与预期存储桶拥有者不匹配，则请求将失败。

如果 Bea 使用存储桶拥有者条件，则之前描述的场景不会导致 Bea 的应用程序无意中写入测试存储桶。她的应用程序在步骤 3 中发出的请求将失败，并显示 `Access Denied` 错误消息。通过使用存储桶拥有者条件，Bea 帮助消除意外与错误 AWS 账户 中的存储桶交互的风险。

验证存储桶拥有者

要使用存储桶拥有者条件，请在请求中包含一个参数，用于指定预期存储桶拥有者。大多数 S3 操作仅涉及单个存储桶，并且只需此单个参数即可使用存储桶拥有者条件。对于 `CopyObject` 操作，此第一个参数指定目标存储桶的预期拥有者，您包含第二个参数来指定源存储桶的预期拥有者。

当您发出包含存储桶拥有者条件参数的请求时，S3 会在处理该请求之前根据指定参数检查存储桶拥有者的账户 ID。如果参数与存储桶拥有者的账户 ID 匹配，S3 将处理该请求。如果参数与存储桶拥有者的账户 ID 不匹配，则该请求将失败并显示 `Access Denied` 错误消息。

您可以通过 AWS Command Line Interface（AWS CLI）、AWS SDK 和 Amazon S3 REST API 来使用存储桶拥有者条件。通过 AWS CLI 和 Amazon S3 REST API 使用存储桶拥有者条件时，请使用以下参数名称。

访问方法	用于非复制操作的参数	复制操作源参数	复制操作目标参数
AWS CLI	`--expected-bucket-owner`	`--expected-source-bucket-owner`	`--expected-bucket-owner`
Amazon S3 REST API	`x-amz-expected-bucket-owner` 标头	`x-amz-source-expected-bucket-owner` 标头	`x-amz-expected-bucket-owner` 标头

通过 AWS SDK 使用存储桶拥有者条件所需的参数名称因语言而异。要确定所需的参数，请参阅所需语言的 SDK 文档。您可以在用于在 AWS 上进行构建的工具中找到 SDK 文档。

示例

以下示例演示如何使用 AWS CLI 或 AWS SDK for Java 2.x 在 Amazon S3 中实施存储桶拥有者条件。

Example

示例：上传对象

以下示例将对象上传到 S3 存储桶*amzn-s3-demo-bucket1*，使用存储桶拥有者条件来确保 *amzn-s3-demo-bucket1* 由 AWS 账户 111122223333 拥有。

AWS CLI

```
aws s3api put-object \
                --bucket amzn-s3-demo-bucket1 --key exampleobject --
body example_file.txt \
                --expected-bucket-owner 111122223333
```

AWS SDK for Java 2.x

```
public void putObjectExample() {
    S3Client s3Client = S3Client.create();;
    PutObjectRequest request = PutObjectRequest.builder()
            .bucket("amzn-s3-demo-bucket1")
            .key("exampleobject")
```

```
            .expectedBucketOwner("111122223333")
            .build();
    Path path = Paths.get("example_file.txt");
    s3Client.putObject(request, path);
}
```

Example

示例：复制对象

以下示例将对象 object1 从 S3 存储桶*amzn-s3-demo-bucket1* 复制到 S3 存储桶*amzn-s3-demo-bucket2*。根据下表，它使用存储桶拥有者条件来确保存储桶由预期账户拥有。

存储桶	预期拥有者
amzn-s3-demo-bucket1	111122223333
amzn-s3-demo-bucket2	444455556666

AWS CLI

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket1/object1 \
                            --bucket amzn-s3-demo-bucket2 --key object1copy \
                            --expected-source-bucket-owner 111122223333 --expected-
bucket-owner 444455556666
```

AWS SDK for Java 2.x

```
public void copyObjectExample() {
        S3Client s3Client = S3Client.create();
        CopyObjectRequest request = CopyObjectRequest.builder()
                .copySource("amzn-s3-demo-bucket1/object1")
                .destinationBucket("amzn-s3-demo-bucket2")
                .destinationKey("object1copy")
                .expectedSourceBucketOwner("111122223333")
                .expectedBucketOwner("444455556666")
                .build();
        s3Client.copyObject(request);
    }
```

Example

示例：检索存储桶策略

以下示例检索 S3 存储桶*amzn-s3-demo-bucket1* 的访问策略，使用存储桶拥有者条件来确保 *amzn-s3-demo-bucket1* 由 AWS 账户 111122223333 拥有。

AWS CLI

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket1 --expected-bucket-
owner 111122223333
```

AWS SDK for Java 2.x

```
public void getBucketPolicyExample() {
    S3Client s3Client = S3Client.create();
    GetBucketPolicyRequest request = GetBucketPolicyRequest.builder()
            .bucket("amzn-s3-demo-bucket1")
            .expectedBucketOwner("111122223333")
            .build();
    try {
        GetBucketPolicyResponse response = s3Client.getBucketPolicy(request);
    }
    catch (S3Exception e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    }
}
```

限制和局限性

Amazon S3 存储桶拥有者条件具有以下限制和局限性：

- 存储桶拥有者条件参数的值必须是 AWS 账户 ID（12 位字母数字值）。不支持服务主体。
- 存储桶拥有者条件不适用于 CreateBucket、ListBuckets 或 AWS S3 控制中包含的任何操作。Amazon S3 将忽略对这些操作的请求中包含的任何存储桶拥有者条件参数。
- 存储桶拥有者条件仅验证在验证参数中指定的账户是否拥有该存储桶。存储桶拥有者条件不检查存储桶的配置。它也不能保证存储桶的配置符合任何特定条件或与任何过去的状态匹配。

为您的存储桶控制对象所有权和禁用 ACL。

S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权并禁用或启用访问控制列表（ACL）。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对数据的访问权限。

Amazon S3 中的大多数现代使用案例不再需要使用 ACL，我们建议您将 ACL 保持为禁用状态，除非有必须单独控制每个对象的访问权限的特殊情况。禁用 ACL 后，您可以使用策略来更轻松地控制对存储桶中每个对象的访问权限，无论是谁将对象上传到存储桶。

对象所有权有三个设置，您可以使用它来控制上传到存储桶的对象的所有权，并禁用或启用 ACL：

已禁用 ACL

- 强制存储桶拥有者（默认）– ACL 已禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响 S3 存储桶中对数据的权限。存储桶使用策略来定义访问控制。

已启用 ACL

- Bucket owner preferred（首选存储桶拥有者）— 存储桶拥有者拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 写入存储桶的新对象。
- 对象编写者— 该 AWS 账户上传对象拥有该对象，对其拥有完全控制权，并且可以通过 ACL 授予其他用户访问该对象的权限。

对于 S3 中的大多数现代使用案例，我们建议您通过应用强制存储桶拥有者设置，并根据需要使用存储桶策略与账户外的用户共享数据，来将 ACL 保持为禁用状态。这种方法可简化权限管理。您可以在新创建的存储桶和现有存储桶上禁用 ACL。对于新创建的存储桶，默认情况下 ACL 处于禁用状态。如果现有存储桶中已有对象，在禁用 ACL 后，对象和存储桶 ACL 将不再是访问评估的一部分，并且根据策略授予或拒绝访问权限。对于现有存储桶，您可以在禁用 ACL 后随时重新启用它们，并恢复原先存在的存储桶和对象 ACL。

在禁用 ACL 之前，我们建议您查看存储桶策略，以确保它涵盖了您打算在账户外授予对存储桶访问权限的所有方式。禁用 ACL 后，您的存储桶仅接受未指定 ACL 的 PUT 请求或具有存储桶拥有者完全控制 ACL 的 PUT 请求，例如 `bucket-owner-full-control` 标准 ACL 或以 XML 表示的此 ACL 的等效形式。支持存储桶拥有者完全控制 ACL 的现有应用程序没有影响。包含其他 ACL 的 PUT 请求（例如，向某些 AWS 账户的自定义授权）失败并返回带有错误代码 `AccessControlListNotSupported` 的 `400` 错误。

反之，具有首选存储桶拥有者设置的存储桶将继续接受和遵守存储桶和对象 ACL。有了这个设置，用 `bucket-owner-full-control` 标准的 ACL 编写的新对象将自动归存储桶拥有者而不是对象编写者所有。所有其他 ACL 行为都保持不变。为了要求所有 Amazon S3 PUT 操作都包含 `bucket-owner-full-control` 标准的 ACL，您可以添加一个存储桶策略，该策略只允许使用该 ACL 上传对象。

要查看哪些对象所有权设置应用于您的存储桶，您可以使用 Amazon S3 Storage Lens 存储统计管理工具指标。S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。有关更多信息，请参阅使用 S3 Storage Lens 存储统计管理工具查找对象所有权设置。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

对象所有权设置

此表显示了每个对象所有权设置对 ACL、对象、Object Ownership 和对象上传的影响。

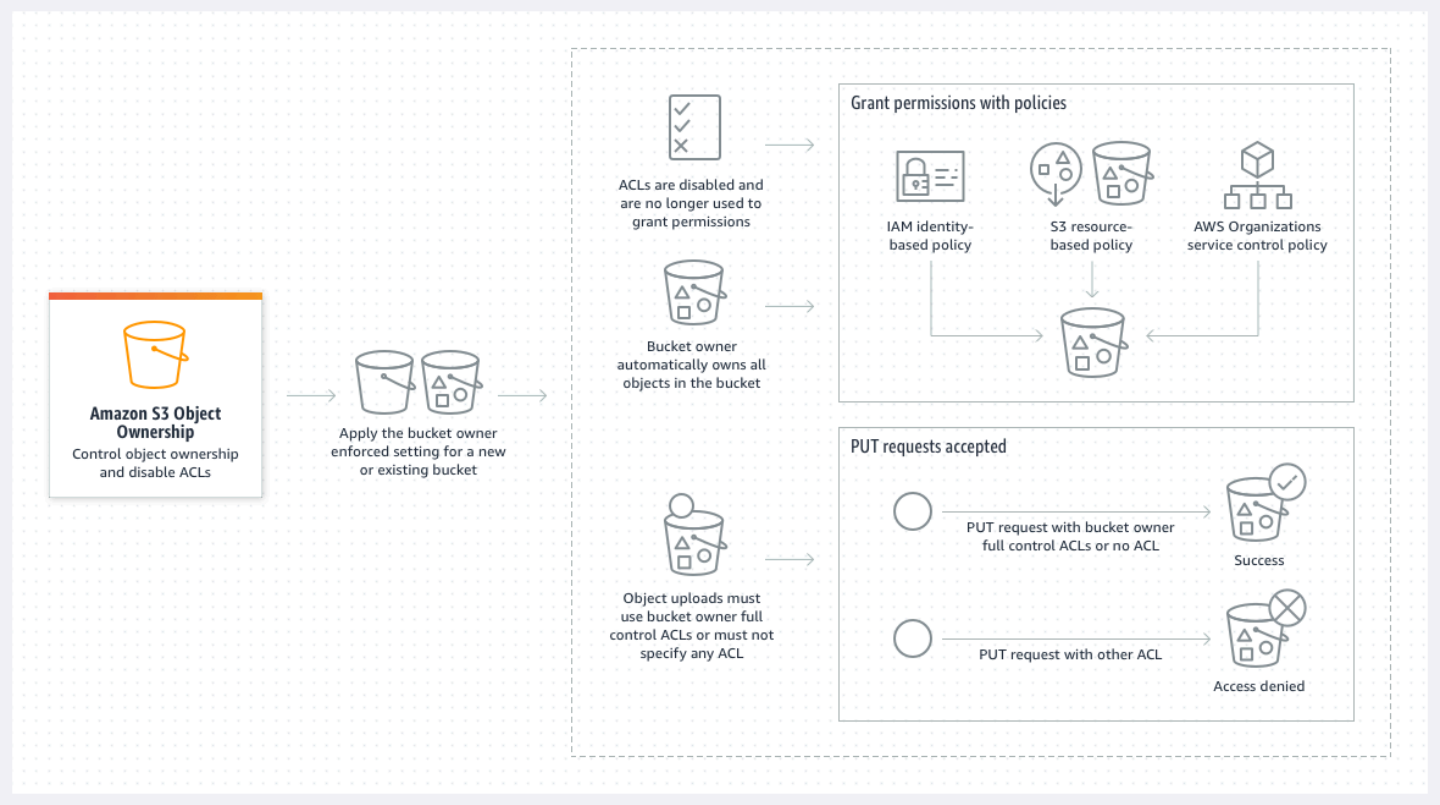
设置	适用于	对 Object Ownership 的影响	对 ACL 的影响	已接受上传
强制存储桶拥有者（默认）	所有新的和现有对象	存储桶拥有者拥有每个对象。	ACL 已禁用，不再影响存储桶的访问权限。设置或更新 ACL 的请求失败。然而，我们仍然支持读取 ACL 的请求。 存储桶拥有者拥有完全所有权和控制	使用存储桶拥有者完全控制 ACL 进行上传或未指定 ACL 的上传

设置	适用于	对 Object Ownership 的影响	对 ACL 的影响	已接受上传
			对象编写者不再拥有完全的所有权和控制权。	
首选存储桶拥有者	新对象	如果对象上传包含 `bucket-owner-full-control` 标准 ACL，存储桶拥有者拥有该对象。 使用其他 ACL 上传的对象归写入账户所有。	ACL 可以更新并可以授予权限。 如果对象上传包含 `bucket-owner-full-control` 标准 ACL，存储桶拥有者具有完全控制权限，对象编写者不再具有完全控制权限。	所有上传项
对象编写者	新对象	对象编写者拥有该对象。	ACL 可以更新并可以授予权限。 对象编写者具有完全控制权限。	所有上传项

通过禁用 ACL 引起的更改

当应用对象所有权的强制存储桶拥有者设置时，将禁用 ACL，并且您将自动拥有并完全控制存储桶中的每个对象，而无需执行任何其他操作。强制存储桶拥有者是所有新创建的存储桶的默认设置。应用强制存储桶拥有者设置后，您将看到三个变化：

- 所有存储桶 ACL 和对象 ACL 都被禁用，这将为您提供作为存储桶拥有者的完全访问权限。当您对存储桶或对象执行读取 ACL 请求时，您将看到仅向存储桶拥有者授予完全访问权限。
- 作为存储桶拥有者，您自动拥有并完全控制存储桶中的每个对象。
- ACL 不再影响存储桶的访问权限。因此，您的数据访问控制基于策略，如 IAM 策略、S3 存储桶策略、VPC 端点策略和 Organizations SCP。



如果您使用 S3 版本控制，则存储桶拥有者拥有并完全控制存储桶中的所有对象版本。应用强制存储桶拥有者设置不会添加对象的新版本。

只有在新对象使用存储桶拥有者完全控制 ACL 或未指定 ACL 时，才能将新对象上传到存储桶。如果指定任何其他 ACL，对象上传将失败。有关更多信息，请参阅 故障排除。

因为以下示例中使用 AWS Command Line Interface（AWS CLI）的 `PutObject` 操作包含 `bucket-owner-full-control` 固有的 ACL，所以目标可以上传至 ACL 禁用的存储桶中。

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key key-name --body path-to-file --
acl bucket-owner-full-control
```

因为以下 `PutObject` 操作没有指定 ACL，对于禁用 ACL 的存储桶，它也会成功。

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key key-name --body path-to-file
```

> **Note**
>
> 如果其他 AWS 账户 在上传后需要对象的访问权限，您必须通过存储桶策略向这些账户授予其他权限。有关更多信息，请参阅 演练：使用策略管理针对 Amazon S3 资源的访问权限。

重新启用 ACL

您可以随时从强制存储桶拥有者设置更改为另一个对象所有权设置来重新启用 ACL。如果在应用强制存储桶拥有者设置之前使用了对象 ACL 进行权限管理，并且您没有将这些对象 ACL 权限迁移到存储桶策略中，则在重新启用 ACL 后，这些权限将恢复。此外，在应用强制存储桶拥有者设置时写入存储桶的对象仍然归存储桶拥有者所有。

例如，如果您从强制存储桶拥有者设置更改回对象编写者设置，则作为存储桶拥有者，您将不再拥有和完全控制以前由其他 AWS 账户拥有的对象。反之，上传账户将再次拥有这些对象。其他账户拥有的对象使用 ACL 获取权限，因此您不能使用策略向这些对象授予权限。但是，您作为存储桶拥有者，在应用强制存储桶拥有者设置时，仍然拥有写入存储桶的任何对象。这些对象不归对象编写者所有，即使您重新启用 ACL 也是如此。

有关使用 AWS Management Console、AWS Command Line Interface（CLI）、REST API 或 AWS SDK 启用和管理 ACL 的说明，请参阅配置 ACL。

禁用 ACL 的先决条件

在为现有存储桶禁用 ACL 之前，请先完成以下先决条件。

查看存储桶和对象 ACL 并迁移 ACL 权限

禁用 ACL 时，存储桶和对象 ACL 授予的权限不再影响访问。在禁用 ACL 之前，请查看存储桶和对象 ACL。

如果您的存储桶 ACL 向账户以外的其他人授予读取或写入权限，则必须将这些权限迁移到存储桶策略，然后才能应用强制存储桶拥有者设置。如果您不迁移在账户之外授予读取或写入访问权限的存储桶 ACL，则应用强制存储桶拥有者设置的请求将失败，并返回 InvalidBucketAclWithObjectOwnership 错误代码。

例如，如果要为接收服务器访问日志的存储桶禁用 ACL，则必须将 S3 日志交付组的存储桶 ACL 权限迁移到存储桶策略中的日志记录服务主体。有关更多信息，请参阅 为服务器访问日志记录授予对 S3 日志传输组的访问权限。

如果您希望对象编写者保持对其上传的对象的完全控制权，则对象编写者是您使用案例的最佳对象所有权设置。如果您想在单个对象级别控制访问权限，那么首选存储桶拥有者是最佳选择。这些用例不常见。

要查看 ACL 并将 ACL 权限迁移到存储桶策略，请参阅 禁用 ACL 的先决条件。

识别需要 ACL 进行授权的请求

要识别需要 ACL 进行授权的 Amazon S3 请求，您可以使用 Amazon S3 服务器访问日志或 AWS CloudTrail 中的 `aclRequired` 值。如果请求需要 ACL 进行授权，或者如果您具有指定 ACL 的 PUT 请求，则字符串为 `Yes`。如果不需要 ACL，或者您正在设置 `bucket-owner-full-control` 标准 ACL，或者如果您的存储桶策略允许请求，则 Amazon S3 服务器访问日志中的 `aclRequired` 值字符串为“-”，但 CloudTrail 中不存在该值字符串。有关预期 `aclRequired` 值的更多信息，请参阅常见 Amazon S3 请求的 `aclRequired` 值。

如果您的 `PutBucketAcl` 或 `PutObjectAcl` 请求的标头授予基于 ACL 的权限（`bucket-owner-full-control` 标准 ACL 除外），则必须先删除这些标头，然后才能禁用 ACL。否则，您的请求将失败。

对于需要 ACL 进行授权的所有其他请求，请将这些 ACL 权限迁移到存储桶策略。然后，在启用强制存储桶拥有者设置之前，删除所有存储桶 ACL。

> **Note**
>
> 请勿删除对象 ACL。否则，依赖于对象 ACL 获取权限的应用程序将失去访问权限。

如果您发现没有请求需要 ACL 进行授权，则可以继续禁用 ACL。有关识别请求的更多信息，请参阅使用 Amazon S3 服务器访问日志来确定请求和使用 CloudTrail 识别 Amazon S3 请求。

查看和更新使用 ACL 相关条件密钥的存储桶策略

应用强制存储桶拥有者设置以禁用 ACL 后，只有在请求使用存储桶拥有者完全控制 ACL 或未指定 ACL 的情况下，才能将新对象上传到存储桶。在禁用 ACL 之前，请查看存储桶策略以了解 ACL 相关的条件密钥。

如果您的存储桶策略使用 ACL 相关的条件密钥来要求 `bucket-owner-full-control` 储存的 ACL（例如，`s3:x-amz-acl`），您无需更新存储桶策略。以下存储桶策略使用 `s3:x-amz-acl` 以要求 `bucket-owner-full-control` 储存的 ACL 用于 S3 `PutObject` 请求。这项策略仍旧要求对象编写者指定 `bucket-owner-full-control` 标准的 ACL。但是，禁用 ACL 的存储桶仍然接受此 ACL，因此请求将继续成功，无需客户端更改。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "Only allow writes to my bucket with bucket owner full control",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

但是，如果您的存储桶策略使用的是要求不同 ACL 的 ACL 相关条件密钥，则必须移除此条件密钥。此示例存储桶策略要求 public-read ACL 用于 S3 PutObject 请求，因此必须在禁用 ACL 之前进行更新。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with public read access",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "public-read"
                }
            }
```

```
        }
    ]
}
```

Object Ownership 权限

要应用、更新或删除存储桶的对象所有权设置，您需要 `s3:PutBucketOwnershipControls` 权限。要返回存储桶的对象所有权设置，您需要 `s3:GetBucketOwnershipControls` 权限。有关更多信息，请参阅 在创建存储桶时设置对象所有权 和 查看 S3 存储桶的对象所有权设置。

禁用所有新存储桶的 ACL

默认情况下，所有新存储桶都是在应用强制存储桶拥有者设置的情况下创建的，且 ACL 处于禁用状态。我们建议将 ACL 保持为禁用状态。作为一般规则，我们建议您使用基于 S3 资源的策略（存储桶策略和接入点策略）或 IAM 策略进行访问控制，而不是 ACL。策略是一种经过简化，且更灵活的访问控制选项。借助存储桶策略和接入点策略，您可以定义广泛适用于针对 Amazon S3 资源的所有请求的规则。

复制和 Object Ownership

当您使用 S3 复制，并且源存储桶和目标存储桶由不同的 AWS 账户拥有时，您可以禁用 ACL（对于对象所有权应用强制存储桶拥有者设置），以将副本拥有权更改为拥有目标存储桶的 AWS 账户。此设置模拟现有的所有者覆盖行为，而无需 `s3:ObjectOwnerOverrideToBucketOwner` 权限。在采用强制存储桶拥有者设置的情况下复制到目标存储桶的所有对象都归目标存储桶拥有者所有。有关复制配置的所有者覆盖选项的更多信息，请参阅 更改副本拥有者。

设置对象所有权

您可以使用 Amazon S3 控制台、AWS CLI、AWS SDK、Amazon S3 REST API 或 AWS CloudFormation 应用对象所有权设置。以下 REST API 和 AWS CLI 命令支持 Object Ownership：

REST API	AWS CLI	描述
PutBucketOwnershipControls	put-bucket-ownership-controls	为现有 S3 存储桶创建或修改对象所有权设置。
CreateBucket	create-bucket	使用创建存储桶 `x-amz-object-ownership` 请求标头以指定对象所有权设置。

REST API	AWS CLI	描述
GetBucketOwnershipControls	get-bucket-ownership-controls	检索 Amazon S3 存储桶的对象所有权设置。
DeleteBucketOwnershipControls	delete-bucket-ownership-controls	删除 Amazon S3 存储桶的对象所有权设置。

有关应用和使用 Object Ownership 设置的更多信息，请参阅以下主题。

主题

- 禁用 ACL 的先决条件
- 在创建存储桶时设置对象所有权
- 为现有存储桶设置对象所有权
- 查看 S3 存储桶的对象所有权设置
- 禁用所有新存储桶的 ACL 并强制执行 Object Ownership
- 故障排除

禁用 ACL 的先决条件

如果您的存储桶 ACL 授予 AWS 账户 以外的访问权限，在禁用 ACL 之前，您必须将存储桶 ACL 权限迁移到存储桶策略中，然后将存储桶 ACL 重置为默认私有 ACL。如果不迁移这些存储桶 ACL，则应用强制存储桶拥有者设置来禁用 ACL 的请求将失败，并返回 InvalidBucketAclWithObjectOwnership 错误代码。我们还建议您查看对象 ACL 权限并将其迁移到存储桶策略中。有关其他所建议的先决条件的更多信息，请参阅 禁用 ACL 的先决条件。

您现有的每个存储桶和对象 ACL 在 IAM 策略中都有等效的。下面的存储桶策略示例向您演示存储桶和对象 ACL 的 `READ` 和 `WRITE` 权限如何映射到 IAM 权限。有关如何将每个 ACL 转换为 IAM 权限的更多信息，请参阅 ACL 权限和访问策略权限的映射。

要查看 ACL 权限并将其迁移至存储桶策略，请参阅以下主题。

主题

- 存储桶策略示例
- 使用 S3 控制台查看和迁移 ACL 权限

- 使用 AWS CLI 查看和迁移 ACL 权限
- 示例演练

存储桶策略示例

这些示例存储桶策略将为您显示如何针对第三方 AWS 账户 将 READ 和 WRITE 存储桶和对象 ACL 权限迁移至存储桶策略，对于策略，READ_ACP 和 WRITE_ACP ACL 相关性较小，这是因为其授予 ACL 相关的权限（s3:GetBucketAcl、s3:GetObjectAcl、s3:PutBucketAcl 和s3:PutObjectAcl）。

Example – 存储桶的 **READ** ACL

如果您的存储桶具有 READ ACL 来向 AWS 账户 *111122223333* 授予列出存储桶内容的权限，则您可以编写授予对存储桶的 s3:ListBucket、s3:ListBucketVersions、s3:ListBucketMultipartUploads 权限的存储桶策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Permission to list the objects in a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [

                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:ListBucket",
                "s3:ListBucketVersions",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
        }
    ]
}
```

Example – 存储桶中每个对象的 **READ** ACL

如果存储桶中的每个对象都有向 AWS 账户 *111122223333* 授予访问权限的 READ ACL，则您可以编写存储桶策略，以便为存储桶中的每个对象授予对此账户的 `s3:GetObject` 和 `s3:GetObjectVersion` 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Read permission for every object in a bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

此示例资源元素授予对特定对象的访问权限。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/OBJECT-KEY"
```

Example – 授予将对象写入存储桶的权限的 **WRITE** ACL

如果您的存储桶具有 WRITE ACL 以向 AWS 账户 *111122223333* 授予将对象写入存储桶的权限，则您可以编写授予对存储桶的 `s3:PutObject` 权限的存储桶策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Permission to write objects to a bucket",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
    ]
}
```

使用 S3 控制台查看和迁移 ACL 权限

查看存储桶的 ACL 权限

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择存储桶名称。
3. 选择 Permissions（权限）选项卡。
4. 在 Access control list（ACL）（访问控制列表）中，查看您的存储桶 ACL 权限。

查看对象的 ACL 权限

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket（存储桶）列表中，选择包含对象的存储桶名称。
3. 在 Objects（对象）列表中，请选择对象名称。
4. 选择 Permissions（权限）选项卡。
5. 在 Access control list (ACL)（访问控制列表）中，查看您的对象 ACL 权限。

要迁移 ACL 权限并更新存储桶 ACL

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择存储桶名称。

3. 在 Permissions（权限）标签页中，在 Bucket policy（存储桶策略）下，请选择 Edit（编辑）。
4. 在 Policy（策略）框中添加或更新存储桶策略。

   有关示例存储桶策略，请参阅 存储桶策略示例 和 示例演练。
5. 选择 Save changes（保存更改）。
6. 更新存储桶 ACL以移除授予其他组或 AWS 账户 的 ACL。
7. 为对象所有权应用强制存储桶拥有者设置。

使用 AWS CLI 查看和迁移 ACL 权限

1. 要返回存储桶的存储桶 ACL，请使用 get-bucket-acl AWS CLI 命令：

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

例如，此存储桶 ACL 授予 WRITE 和 READ 访问第三方账户。在此 ACL 中，第三方账户由 canonical user ID（规范的用户 ID）识别。要应用强制存储桶拥有者设置并禁用 ACL，您必须将第三方账户的这些权限迁移到存储桶策略。

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
                "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
                "Type": "CanonicalUser"
            },
```

```
            "Permission": "READ"
        },
        {
            "Grantee": {
                "DisplayName": "THIRD-PARTY-EXAMPLE-ACCOUNT",
                "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
        }
    ]
}
```

有关其他示例 ACL，请参阅 示例演练。

2. 将存储桶 ACL 权限迁移到存储桶策略：

此示例存储桶策略为第三方账户授予 s3:PutObject 和 s3:ListBucket 权限。在存储桶策略中，第三方账户由 AWS 账户 ID（*111122223333*）标识。

```
aws s3api put-bucket-policy --bucket DOC-EXAMPLE-BUCKET --policy file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PolicyForCrossAccountAllowUpload",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
             "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
             "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
        }
```

```
    ]
}
```

有关更多示例存储桶策略，请参阅 存储桶策略示例 和 示例演练。

3. 要返回特定对象的 ACL，请使用 get-object-acl AWS CLI 命令。

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key EXAMPLE-OBJECT-KEY
```

4. 如果需要，请将对象 ACL 权限迁移到存储桶策略中。

此示例资源元素授予对存储桶策略中特定对象的访问权限。

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/EXAMPLE-OBJECT-KEY"
```

5. 将存储桶的 ACL 重置为默认的 ACL。

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

6. 为对象所有权应用强制存储桶拥有者设置。

示例演练

以下示例向您演示如何针对特定使用案例将 ACL 权限迁移到存储桶策略。

主题

- 为服务器访问日志记录授予对 S3 日志传输组的访问权限
- 授予存储桶中对象的公共读取访问权限。
- 授予 Amazon ElastiCache（Redis OSS）对 Amazon S3 存储桶的访问权限

为服务器访问日志记录授予对 S3 日志传输组的访问权限

如果要应用强制存储桶拥有者设置以便对服务器访问日志记录目标存储桶（也称作目标存储桶）禁用 ACL，必须将 S3 日志传输组的存储桶 ACL 权限迁移到存储桶策略中的日志记录服务主体（logging.s3.amazonaws.com）。有关日志交付权限的更多信息，请参阅 日志传输的权限。

此存储桶 ACL 授予 WRITE 和 READ_ACP 访问 S3 日志传输组的权限：

```
{
    "Owner": {
```

```
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "Type": "CanonicalUser",
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

将 S3 日志传输组的存储桶 ACL 权限迁移到存储桶策略中的日志录入服务主体

1. 将以下存储桶策略添加到目标存储桶，并替换示例值。

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json

policy.json:      {
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/EXAMPLE-LOGGING-PREFIX*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::SOURCE-BUCKET-NAME"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

2. 将目标存储桶的 ACL 重置为默认 ACL。

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

3. 为目标存储桶的对象所有权应用强制存储桶拥有者设置。

授予存储桶中对象的公共读取访问权限。

如果您的对象 ACL 授予对存储桶中所有对象的公共读取访问权限，则可以将这些 ACL 权限迁移到存储桶策略。

此对象 ACL 授予对存储桶中对象的公共读取访问权限：

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
```

```
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

要将公共读取 ACL 权限迁移到存储桶策略

1. 要向存储桶中的所有对象授予公有读取权限，请添加以下存储桶策略，替换示例值。

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy
 file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

要授予对存储桶策略中特定对象的公共访问权限，请使用以下 Resource 元素的格式。

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/OBJECT-KEY"
```

要使用特定的前缀授予对所有对象的公共访问权，请使用以下 Resource 元素的格式。

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/PREFIX/*"
```

2. 为对象所有权应用强制存储桶拥有者设置。

授予 Amazon ElastiCache（Redis OSS）对 Amazon S3 存储桶的访问权限

您可以将 ElastiCache（Redis OSS）备份导出到 S3 存储桶中，这样您就可以从 ElastiCache 外部访问备份。要将备份导出到 S3 存储桶，您必须授予 ElastiCache 权限，以便将快照复制到存储桶。如果您已在存储桶 ACL 中授予了对 ElastiCache 的权限，则必须在应用强制存储桶拥有者设置以禁用 ACL 之前将这些权限迁移到存储桶策略。有关更多信息，请参阅 Amazon ElastiCache User Guide（Amazon ElastiCache 用户指南）中的 Grant ElastiCache access to your Amazon S3 存储桶（授予 ElastiCache 访问 Amazon S3 存储桶的权限）。

以下示例显示了向 ElastiCache 授予权限的存储桶 ACL 权限。

```
{
    "Owner": {
        "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "DOC-EXAMPLE-ACCOUNT-OWNER",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
```

```
            },
            "Permission": "READ"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "DisplayName": "aws-scs-s3-readonly",
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

将用于 ElastiCache（Redis OSS）的存储桶 ACL 权限迁移到存储桶策略

1. 将以下存储桶策略添加到存储桶，替换示例值。

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy
 file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "Region.elasticache-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
```

```
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

2. 将存储桶的 ACL 重置为默认的 ACL：

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

3. 为对象所有权应用强制存储桶拥有者设置。

在创建存储桶时设置对象所有权

创建存储桶时，您可以配置 S3 对象所有权。要为现有存储桶设置对象所有权，请参阅 为现有存储桶设置对象所有权。

S3 对象所有权是 Amazon S3 存储桶级设置，您可以使用它禁用 access control lists（ACLs）（访问控制列表 ACL），并获取存储桶中每个对象的所有权，从而简化了对存储在 Amazon S3 中的数据的访问管理。默认情况下，S3 对象所有权设为强制存储桶拥有者设置，并且对于新存储桶禁用 ACL。禁用 ACL 后，存储桶拥有者拥有存储桶中的每个对象，并使用访问管理策略来专门管理对数据的访问权限。我们建议您将 ACL 保持为禁用状态，除非有必须单独控制每个对象的访问权限的特殊情况。

对象所有权有三个设置，您可以使用它来控制上传到存储桶的对象的所有权，并禁用或启用 ACL：

已禁用 ACL

- 强制存储桶拥有者（默认）– ACL 已禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响 S3 存储桶中对数据的权限。存储桶使用策略来定义访问控制。

已启用 ACL

- Bucket owner preferred（首选存储桶拥有者）— 存储桶拥有者拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 写入存储桶的新对象。
- 对象编写者— 该 AWS 账户上传对象拥有该对象，对其拥有完全控制权，并且可以通过 ACL 授予其他用户访问该对象的权限。

权限：要应用 Bucket owner enforced（强制存储桶拥有者）设置或 Bucket owner preferred（首选存储桶拥有者）设置，必须有以下权限：`s3:CreateBucket` 和 `s3:PutBucketOwnershipControls`。在应用了 Object writer（对象编写者）设置的情况下创建存储桶时，不需要额外的权限。有关 Amazon S3 权限的更多信息，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

> **Important**
>
> Amazon S3 中的大多数现代使用案例不再需要使用 ACL，我们建议您禁用 ACL，除非在需要单独控制每个对象的访问的异常情况下。使用对象所有权，您可以禁用 ACL 并依赖策略进行访问控制。禁用 ACL 时，您可以轻松维护具有通过不同的 AWS 账户上传的对象的存储桶。作为存储桶拥有者，您拥有存储桶中的所有对象，并可以使用策略管理对它们的访问。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要在其中创建存储桶的区域。

   > **Note**
   >
   > 要最大程度地减少延迟和成本以及满足法规要求，请选择一个靠近您的区域。在某一区域存储的对象将一直留在该区域，除非您特意将其转移到其他区域。有关 Amazon S3 AWS 区域的列表，请参阅《Amazon Web Services 一般参考》中的 AWS 服务 端点。

3. 在左侧导航窗格中，选择存储桶。
4. 选择创建存储桶。

   此时将打开创建存储桶页面。

5. 在常规配置下，查看将在其中创建存储桶的 AWS 区域。
6. 在存储桶类型下，选择通用。
7. 对于存储桶名称，请输入存储桶的名称。

   存储桶名称必须满足以下要求：

   - 在分区中是唯一的。分区是一组区域。AWS 目前有三个分区：aws（标准区域）、aws-cn（中国区域）和 aws-us-gov（AWS GovCloud (US) Regions）。
   - 长度必须介于 3 到 63 个字符之间。
   - 只能由小写字母、数字、句点（.）和连字符（-）组成。为了获得最佳兼容性，我们建议您避免在存储桶名称中使用句点（.），但仅用于静态网站托管的存储桶除外。
   - 以字母或数字开头和结尾。

   创建存储桶后，便无法再更改其名称。有关给存储桶命名的更多信息，请参阅存储桶命名规则。

   > **Important**
   >
   > 避免在存储桶名称中包含敏感信息，如账号。存储桶名称会显示在指向存储桶中的对象的 URL 中。

8. AWS Management Console可让您将现有存储桶的设置复制到新存储桶。如果您不想复制现有存储桶的设置，请跳到下一步。

   > **Note**
   >
   > 此选项：
   >
   > - 在 AWS CLI 中不可用，仅在控制台中可用
   > - 不可用于目录存储桶
   > - 不会将存储桶策略从现有存储桶复制到新存储桶

   要复制现有存储桶的设置，请在从现有存储桶复制设置下，选择选择存储桶。将打开选择存储桶窗口。找到包含您要复制的设置的存储桶，然后选择选择存储桶。选择存储桶窗口关闭，创建存储桶窗口重新打开。

在从现有存储桶复制设置下，您现在将看到所选存储桶的名称。您还将看到还原默认值选项，您可以使用该选项移除复制的存储桶设置。在创建存储桶页面上查看其余存储桶设置。您将看到它们现在与您选择的存储桶的设置相匹配。您可以跳到最后一步。

9. 在 Object Ownership（对象所有权）下方，要禁用或启用 ACL，并控制上传到存储桶中的对象的所有权，请选择以下设置之一：

   已禁用 ACL

   - 强制存储桶拥有者（默认）– ACL 已禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响对 S3 存储桶中数据的访问权限。存储桶专门使用策略来定义访问控制。

     默认情况下，ACL 处于禁用状态。Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有必须单独控制每个对象的访问权限的特殊情况。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

   已启用 ACL

   - Bucket owner preferred（首选存储桶拥有者）— 存储桶拥有者拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 写入存储桶的新对象。

     如果应用首选存储桶拥有者设置，以要求所有 Amazon S3 上传都包含 `bucket-owner-full-control` 标准 ACL，则可以添加存储桶策略，该策略只允许使用此 ACL 上传对象。
   - 对象编写者— 该 AWS 账户上传对象拥有该对象，对其拥有完全控制权，并且可以通过 ACL 授予其他用户访问该对象的权限。

   > **Note**
   >
   > 默认设置为强制存储桶拥有者。要应用默认设置并将 ACL 保持为禁用状态，只需要 `s3:CreateBucket` 权限。要启用 ACL，您必须具有 `s3:PutBucketOwnershipControls` 权限。

10. 在此存储桶的屏蔽公共访问权限设置中，请选择要应用于存储桶的屏蔽公共访问权限设置。

    默认情况下，启用所有四个“屏蔽公共访问权限”设置。我们建议您将所有设置保持为启用状态，除非您知道您需要为您的特定使用案例关闭其中一个或多个设置。有关屏蔽公共访问权限的更多信息，请参阅阻止对您的 Amazon S3 存储的公有访问。

> **Note**
>
> 要启用所有“屏蔽公共访问权限”设置，只需要 `s3:CreateBucket` 权限。要关闭任何“屏蔽公共访问权限”设置，您必须拥有 `s3:PutBucketPublicAccessBlock` 权限。

11. （可选）在 Bucket Versioning（存储桶版本控制）下，您可以选择是否要在存储桶中保留对象的变体。有关版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。

    要在存储桶上禁用或启用版本控制，请选择 Disable（禁用）或 Enable（启用）。

12. （可选）在 Tags（标签）下，您可以选择向存储桶添加标签。标签是用于对存储进行分类的键/值对。

    要添加存储桶标签，请输入 Key（键），并（可选）输入 Value（值），然后选择 Add Tag（添加标签）。

13. 在默认加密下，请选择编辑。

14. 要配置默认加密，请在加密类型下，选择以下选项之一：

    - Amazon S3 托管密钥（SSE-S3）
    - AWS Key Management Service 密钥（SSE-KMS）

    > **Important**
    >
    > 如果您将 SSE-KMS 选项用于默认加密配置，则您将受到 AWS KMS 的每秒请求数（RPS）限额限制。有关 AWS KMS 限额以及如何请求增加限额的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的限额。

    存储桶和新对象使用具有 Amazon S3 托管密钥的服务器端加密进行加密，作为加密配置的基本级别。有关默认加密的更多信息，请参阅为 Amazon S3 存储桶设置默认服务器端加密行为。

    有关使用 Amazon S3 服务器端加密对数据进行加密的更多信息，请参阅使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

15. 如果选择了 AWS Key Management Service 密钥（SSE-KMS），则执行以下操作：

    a. 在 AWS KMS 密钥下，通过以下方式之一指定您的 KMS 密钥：

- 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 密钥中进行选择，并从可用密钥的列表中选择您的 KMS 密钥。

  AWS 托管式密钥 (aws/s3) 和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。

- 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。

- 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

  有关创建 AWS KMS key 的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

> **Important**
>
> 您只能使用与存储桶所在相同的 AWS 区域中可用的 KMS 密钥。Amazon S3 控制台仅列出与存储桶位于同一区域中的前 100 个 KMS 密钥。要使用未列出的 KMS 密钥，您必须输入 KMS 密钥 ARN。如果您希望使用其他账户拥有的 KMS 密钥，则必须首先有权使用该密钥，然后必须输入相应的 KMS 密钥 ARN。有关 KMS 密钥的跨账户权限的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建其他账户可以使用的 KMS 密钥。有关 SSE-KMS 的更多信息，请参阅使用 AWS KMS (SSE-KMS) 指定服务器端加密。
>
> 在 Amazon S3 中使用 AWS KMS key 进行服务器端加密时，您必须选择对称加密 KMS 密钥。Amazon S3 仅支持对称加密 KMS 密钥，而不支持非对称 KMS 密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的确定对称和非对称 KMS 密钥。

有关创建 AWS KMS key 的更多信息，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。有关将 AWS KMS 与 Amazon S3 结合使用的更多信息，请参阅使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。

b. 将存储桶配置为使用 SSE-KMS 进行默认加密时，您还可以启用 S3 存储桶密钥。S3 存储桶密钥可通过减少从 Amazon S3 到 AWS KMS 的请求流量，降低加密成本。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。

   要使用 S3 存储桶密钥，请在 Bucket Key（存储桶密钥）下，选择 Enable（启用）。

16. （可选）如果要启用 S3 对象锁定，请执行以下操作：

    a. 选择高级设置。

    > **Important**
    >
    > 启用对象锁定还会启用存储桶的版本控制。启用后，必须配置对象锁定默认保留和法律保留设置，以保护新对象不被删除或覆盖。

    b. 如果要启用对象锁定，请选择 Enable（启用），阅读出现的警告，并予以确认。

    有关更多信息，请参阅 使用 S3 对象锁定。

    > **Note**
    >
    > 要创建启用对象锁定的存储桶，您必须具有以下权限：`s3:CreateBucket`、`s3:PutBucketVersioning` 和 `s3:PutBucketObjectLockConfiguration`。

17. 请选择创建存储桶。

使用 AWS CLI

要在创建新存储桶时设置对象所有权，请使用带有 `--object-ownership` 参数的 `create-bucket` AWS CLI 命令。

下面的例子使用 AWS CLI 为新存储桶应用了强制存储桶拥有者设置：

```
aws s3api create-bucket --bucket  amzn-s3-demo-bucket --region us-east-1 --object-
ownership BucketOwnerEnforced
```

> **Important**
>
> 如果您在使用 AWS CLI 创建存储桶时未设置对象所有权，则默认设置将为 `ObjectWriter`（启用 ACL）。

使用适用于 Java 的 AWS 软件开发工具包

下面的例子使用 AWS SDK for Java 为新存储桶设置了强制存储桶拥有者设置：

```
    // Build the ObjectOwnership for CreateBucket
    CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
            .build()

     // Send the request to Amazon S3
     s3client.createBucket(createBucketRequest);
```

使用 AWS CloudFormation

要在创建新存储桶时使用 `AWS::S3::Bucket` AWS CloudFormation 资源以设置对象所有权，请参阅《AWS CloudFormation 用户指南》中的 AWS::S3::Bucket 中的 OwnershipControls。

使用 REST API

要为 S3 对象所有权应用强制存储桶拥有者设置，请使用 `x-amz-object-ownership` 请求标头设置为 `BucketOwnerEnforced` 的 `CreateBucket` API 操作。有关信息和示例，请参阅《Amazon Simple Storage Service API 参考》中的 CreateBucket。

后续步骤：在应用对象所有权的强制存储桶拥有者或首选存储桶拥有者设置后，您可以进一步执行以下步骤：

- Bucket owner enforced（强制存储桶拥有者）—通过使用 IAM 或企业策略借助禁用的 ACL 要求创建所有新的存储桶。
- Bucket owner preferred（首选存储桶拥有者）—添加 S3 存储桶策略，要求将所有对象上传到存储桶中都需要 `bucket-owner-full-control` 预定义 ACL。

为现有存储桶设置对象所有权

您可以在现有 S3 存储桶上配置 S3 对象所有权。要在创建存储桶时应用对象所有权，请参阅 在创建存储桶时设置对象所有权。

S3 对象所有权是 Amazon S3 存储桶级设置，您可以使用它禁用 access control lists（ACLs）（访问控制列表 ACL），并获取存储桶中每个对象的所有权，从而简化了对存储在 Amazon S3 中的数据的

访问管理。默认情况下，S3 对象所有权设为强制存储桶拥有者设置，并且对于新存储桶禁用 ACL。禁用 ACL 后，存储桶拥有者拥有存储桶中的每个对象，并使用访问管理策略来专门管理对数据的访问权限。我们建议您将 ACL 保持为禁用状态，除非有必须单独控制每个对象的访问权限的特殊情况。

对象所有权有三个设置，您可以使用它来控制上传到存储桶的对象的所有权，并禁用或启用 ACL：

已禁用 ACL

- 强制存储桶拥有者（默认）– ACL 已禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响 S3 存储桶中对数据的权限。存储桶使用策略来定义访问控制。

已启用 ACL

- Bucket owner preferred（首选存储桶拥有者）— 存储桶拥有者拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 写入存储桶的新对象。
- 对象编写者— 该 AWS 账户上传对象拥有该对象，对其拥有完全控制权，并且可以通过 ACL 授予其他用户访问该对象的权限。

先决条件：在应用强制存储桶拥有者设置以禁用 ACL 之前，您必须将存储桶 ACL 权限迁移到存储桶策略并将存储桶 ACL 重置为默认私有 ACL。我们还建议您将对象 ACL 权限迁移到存储桶策略，并编辑需要存储桶拥有者完全控制 ACL 以外的 ACL 的存储桶策略。有关更多信息，请参阅 禁用 ACL 的先决条件。

Permissions（权限）：要使用此操作，您必须拥有 `s3:PutBucketOwnershipControls` 权限。有关 Amazon S3 权限的更多信息，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要将 S3 对象所有权设置应用到的存储桶的名称。
3. 选择 Permissions（权限）选项卡。
4. 在 Object Ownership（对象所有权）下方，请选择 Edit（编辑）。
5. 在 Object Ownership（对象所有权）下方，要禁用或启用 ACL，并控制上传到存储桶中的对象的所有权，请选择以下设置之一：

已禁用 ACL

- Bucket owner enforced（强制存储桶拥有者）— ACL 被禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响 S3 存储桶中对数据的权限。存储桶使用策略来定义访问控制。

  要通过使用 IAM 或 AWS Organizations 策略借助禁用的 ACL 要求创建所有新的存储桶，请参阅 禁用所有新存储桶的 ACL（强制存储桶拥有者）。

已启用 ACL

- Bucket owner preferred（首选存储桶拥有者）— 存储桶拥有者拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 写入存储桶的新对象。

  如果应用首选存储桶拥有者设置，以要求所有的 Amazon S3 上传都包含 `bucket-owner-full-control` 存储的 ACL，可以 add a bucket policy（添加存储桶策略），该策略只允许对象使用此 ACL 上传。
- 对象编写者— 该 AWS 账户上传对象拥有该对象，对其拥有完全控制权，并且可以通过 ACL 授予其他用户访问该对象的权限。

6. 请选择保存。

使用 AWS CLI

要为现有存储桶应用 Object Ownership 设置，请使用带有 `--ownership-controls` 参数的 `put-bucket-ownership-controls` 命令。所有权的有效值为 `BucketOwnerEnforced`、`BucketOwnerPreferred` 或 `ObjectWriter`。

下面的例子使用 AWS CLI 为现有的存储桶应用强制存储桶拥有者设置：

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
```

有关 `put-bucket-ownership-controls` 的信息，请参阅《AWS Command Line Interface 用户指南》中的 put-bucket-ownership-controls。

使用适用于 Java 的 AWS SDK

此示例使用 AWS SDK for Java 对现有存储桶应用了存储桶拥有者的 `BucketOwnerEnforced` 设置：

```
        // Build the ObjectOwnership for BucketOwnerEnforced
        OwnershipControlsRule rule = OwnershipControlsRule.builder()
               .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
               .build();

        OwnershipControls ownershipControls = OwnershipControls.builder()
                  .rules(rule)
                  .build()

         // Build the PutBucketOwnershipControlsRequest
         PutBucketOwnershipControlsRequest putBucketOwnershipControlsRequest =
               PutBucketOwnershipControlsRequest.builder()
                       .bucket(BUCKET_NAME)
                       .ownershipControls(ownershipControls)
                       .build();

         // Send the request to Amazon S3
         s3client.putBucketOwnershipControls(putBucketOwnershipControlsRequest);
```

使用 AWS CloudFormation

要使用 AWS CloudFormation 以便为现有存储桶应用对象所有权设置，请参阅《AWS CloudFormation 用户指南》中的 AWS::S3::Bucket OwnershipControls。

使用 REST API

要使用 REST API 将 Object Ownership 设置应用于现有 S3 存储桶，请使用 `PutBucketOwnershipControls`。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketOwnershipControls。

后续步骤：在应用对象所有权的强制存储桶拥有者或首选存储桶拥有者设置后，您可以进一步执行以下步骤：

- Bucket owner enforced（强制存储桶拥有者）—通过使用 IAM 或企业策略借助禁用的 ACL 要求创建所有新的存储桶。
- Bucket owner preferred（首选存储桶拥有者）—添加 S3 存储桶策略，要求将所有对象上传到存储桶中都需要 `bucket-owner-full-control` 预定义 ACL。

查看 S3 存储桶的对象所有权设置

S3 对象所有权是 Amazon S3 存储桶级设置，您可以使用它禁用 access control lists（ACLs）（访问控制列表 ACL），并获取存储桶中每个对象的所有权，从而简化了对存储在 Amazon S3 中的数据的访问管理。默认情况下，S3 对象所有权设为强制存储桶拥有者设置，并且对于新存储桶禁用 ACL。禁用 ACL 后，存储桶拥有者拥有存储桶中的每个对象，并使用访问管理策略来专门管理对数据的访问权限。我们建议您将 ACL 保持为禁用状态，除非有必须单独控制每个对象的访问权限的特殊情况。

对象所有权有三个设置，您可以使用它来控制上传到存储桶的对象的所有权，并禁用或启用 ACL：

已禁用 ACL

- 强制存储桶拥有者（默认）– ACL 已禁用，存储桶拥有者自动拥有并完全控制存储桶中的每个对象。ACL 不再影响 S3 存储桶中对数据的权限。存储桶使用策略来定义访问控制。

已启用 ACL

- Bucket owner preferred（首选存储桶拥有者）— 存储桶拥有者拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 写入存储桶的新对象。
- 对象编写者— 该 AWS 账户上传对象拥有该对象，对其拥有完全控制权，并且可以通过 ACL 授予其他用户访问该对象的权限。

查看 Amazon S3 存储桶的对象所有权设置。要为新的存储桶设置对象所有权，请参阅 在创建存储桶时设置对象所有权。要为现有存储桶设置对象所有权，请参阅 为现有存储桶设置对象所有权。

Permissions（权限）：要使用此操作，您必须拥有 `s3:GetBucketOwnershipControls` 权限。有关 Amazon S3 权限的更多信息，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要将对象所有权设置应用到的存储桶的名称。
3. 选择 Permissions（权限）选项卡。
4. 在 Object Ownership（对象所有权）下方，您可以查看存储桶的对象所有权设置。

使用 AWS CLI

要检索 S3 存储桶的 S3 对象所有权设置，请使用 get-bucket-ownership-controls AWS CLI 命令。

```
aws s3api get-bucket-ownership-controls --bucket amzn-s3-demo-bucket
```

使用 REST API

要检索 S3 存储桶的 Object Ownership 设置，请使用 `GetBucketOwnershipControls` API 操作。有关更多信息，请参阅 GetBucketOwnershipControls。

禁用所有新存储桶的 ACL 并强制执行 Object Ownership

建议禁用 Amazon S3 存储桶上的 ACL。可以通过为 S3 对象所有权应用强制存储桶拥有者设置来实现这一点。应用此设置时，ACL 将被禁用，并且您自动拥有并完全控制存储桶中的所有对象。若要求在禁用 ACL 的情况下创建所有新存储桶，请使用 AWS Identity and Access Management（IAM）策略或 AWS Organizations 服务控制策略（SCP），如下一节所述。

要在不禁用 ACL 的情况下强制对新对象的对象所有权，您可以应用“首选存储桶拥有者”设置。当您应用此设置时，我们强烈建议您更新存储桶策略，以要求对存储桶的所有 PUT 请求使用 `bucket-owner-full-control` 标准 ACL。确保还更新客户端，以将 `bucket-owner-full-control` 标准 ACL 从其他账户发送至存储桶。

主题

- 禁用所有新存储桶的 ACL（强制存储桶拥有者）
- 要求 Amazon S3 PUT 操作使用存储桶拥有者完全控制的标准 ACL（首选存储桶拥有者）

禁用所有新存储桶的 ACL（强制存储桶拥有者）

下面的示例 IAM 策略拒绝特定 IAM 用户或角色的 `s3:CreateBucket` 权限，除非为对象所有权应用了强制存储桶拥有者设置。`Condition` 块中的键值对指定 `s3:x-amz-object-ownership` 为其密钥，并将 `BucketOwnerEnforced` 设为其值。换句话说，只有在 IAM 用户为对象所有权设定强制存储桶拥有者设置并禁用 ACL 时，这些用户才能创建存储桶。您还可以将此策略用作 AWS 企业的边界的 SCP。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "RequireBucketOwnerFullControl",
            "Action": "s3:CreateBucket",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-object-ownership": "BucketOwnerEnforced"
                }
            }
        }
    ]
}
```

要求 Amazon S3 **PUT** 操作使用存储桶拥有者完全控制的标准 ACL（首选存储桶拥有者）

通过为对象所有权使用“首选存储桶拥有者”设置，您作为存储桶拥有者，拥有并完全控制其他账户使用 `bucket-owner-full-control` 标准 ACL 向存储桶写入的新对象。但是，如果其他账户在没有 `bucket-owner-full-control` 标准 ACL 的情况下将对象写入存储桶，对象编写者将保持完全控制访问权限。作为存储桶拥有者，只有指定了 `bucket-owner-full-control` 标准的 ACL，才能实现允许写入的存储桶策略。

> **Note**
>
> 如果在应用强制存储桶拥有者设置的情况下禁用了 ACL，则作为存储桶拥有者，您将自动拥有并完全控制存储桶中的所有对象。您无需使用此部分来更新存储桶策略来强制执行存储桶拥有者的对象所有权。

以下存储桶策略指定只有当对象的 ACL 设置为 111122223333 时，账户 *amzn-s3-demo-bucket* 才能将对象上传到 *bucket-owner-full-control*。请务必将 *111122223333* 替换为您的账户，*amzn-s3-demo-bucket* 替换为存储桶的名称。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Only allow writes to my bucket with bucket owner full control",
         "Effect": "Allow",
         "Principal": {
            "AWS": [
               "arn:aws:iam::111122223333:user/ExampleUser"
```

```
            ]
         },
         "Action": [
            "s3:PutObject"
         ],
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
         "Condition": {
            "StringEquals": {
               "s3:x-amz-acl": "bucket-owner-full-control"
            }
         }
      }
   ]
}
```

以下是一个复制操作示例，其中包括使用 AWS Command Line Interface（AWS CLI）的 `bucket-owner-full-control` 标准 ACL。

```
aws s3 cp file.txt s3://amzn-s3-demo-bucket --acl bucket-owner-full-control
```

存储桶策略生效之后，如果客户端不包含 `bucket-owner-full-control` 标准 ACL，则操作将失败，上传者将收到以下错误：

调用 PutObject 操作时发生错误 (AccessDenied)：访问被拒绝。

> **Note**
>
> 如果客户端在上传后需要访问对象，则必须向上传账户授予其他权限。有关授予账户对资源的访问权限的信息，请参阅 演练：使用策略管理针对 Amazon S3 资源的访问权限。

故障排除

在应用 S3 对象所有权的强制存储桶拥有者设置时，访问控制列表（ACL）将被禁用，作为存储桶拥有者的您将自动拥有存储桶中的所有对象。ACL 不再影响存储桶中对象的权限。您可使用策略授予权限。所有 S3 PUT 请求都必须指定 `bucket-owner-full-control` 标准 ACL 或不指定 ACL，否则这些请求将失败。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

如果指定了无效的 ACL 或存储桶 ACL 权限授予 AWS 账户 外部的访问权限，您可能会看到以下错误响应。

AccessControlListNotSupported

为对象所有权应用强制存储桶拥有者设置之后，将禁用 ACL。设置 ACL 或更新 ACL 的请求失败，并显示 400 错误，并返回 AccessControlListNotSupported 错误代码。仍然支持读取 ACL 的请求。读取 ACL 的请求始终返回一个响应，显示对存储桶拥有者的完全控制权。在 PUT 操作中，您必须指定存储桶拥有者完全控制 ACL 或不指定 ACL。否则，您的 PUT 操作将失败。

以下示例 `put-object` AWS CLI 命令包括 `public-read` 标准 ACL。

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key object-key-name --body doc-example-body --acl public-read
```

如果存储桶使用强制存储桶拥有者设置来禁用 ACL，则此操作将失败，上传者会收到以下错误消息：

调用 PutObject 操作时发生错误（AccessControlListNotSupported）：存储桶不允许 ACL

InvalidBucketAclWithObjectOwnership

如果您想应用强制存储桶拥有者设置来禁用 ACL，则存储桶 ACL 必须仅向存储桶拥有者提供完全控制权。您的存储桶 ACL 无法访问外部 AWS 账户 或任何其他团体。例如，如果您的 `CreateBucket` 请求设置强制存储桶拥有者并指定一个存储桶 ACL，用于提供对外部 AWS 账户的访问权限，您的请求会失败并显示 400 错误，并且返回 InvalidBucketAclWithObjectOwnership 错误代码。同样，如果您的 `PutBucketOwnershipControls` 请求对于具有存储桶 ACL（可向其他人授予权限）的存储桶设置强制存储桶拥有者，请求将失败。

Example ：授予存储桶 ACL 公有读取访问权限。

例如，如果现有存储桶 ACL 授予公共读取访问权限，则在将这些 ACL 权限迁移到存储桶策略并将存储桶 ACL 重置为默认私有 ACL 之前，您无法对于对象所有权应用强制存储桶拥有者设置。有关更多信息，请参阅 禁用 ACL 的先决条件。

此示例存储桶 ACL 授予公共读取访问权限：

```
{
    "Owner": {
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
```

```
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

以下示例 put-bucket-ownership-controls AWS CLI 命令对于对象所有权应用强制存储桶拥有者设置：

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls Rules=[{ObjectOwnership=BucketOwnerEnforced}]
```

由于存储桶 ACL 授予公共读取访问权限，因此请求失败并返回以下错误代码：

调用 PutBucketOwnershipControls 操作时发生错误（InvalidBucketAclWithObjectOwnership）：存储桶不能含有带有 ObjectOwnership's BucketOwnerEnforced 设置的 ACL

Amazon S3 中的日志记录和监控

监控是保持 Amazon S3 和您的 AWS 解决方案的可靠性、可用性和性能的重要方面。您应该从 AWS 解决方案的各个部分收集监控数据，以便您可以更轻松地调试多点故障（如果发生）。AWS 提供了多种工具来监控您的 Amazon S3 资源并对潜在事件做出响应。

有关更多信息，请参阅 监控 Amazon S3。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

Amazon CloudWatch 警报

使用 Amazon CloudWatch 警报，您可以在指定时间段内监控某个指标。如果指标超过给定阈值，则会向 Amazon SNS 主题或 AWS Auto Scaling 策略发送通知。CloudWatch 警报将不会调用操作，因为这些操作处于特定状态。而是必须在状态已改变并在指定的若干个时间段内保持不变后才调用。有关更多信息，请参阅 使用 Amazon CloudWatch 监控指标。

AWS CloudTrail 日志

CloudTrail 提供了用户、角色或 AWS 服务在 Amazon S3 中所执行操作的记录。使用 CloudTrail 收集的信息，您可以确定向 Amazon S3 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon S3 API 调用。

Amazon GuardDuty

Amazon GuardDuty 是一项威胁检测服务，可持续监控您的账户、容器、工作负载和 AWS 环境中的数据，以识别 S3 存储桶面临的潜在威胁或安全风险。GuardDuty 还提供了有关所检测到的威胁的丰富上下文信息。GuardDuty 会监控 AWS CloudTrail 管理日志中是否存在威胁，并显示与安全相关的信息。例如，GuardDuty 会将 API 请求的各个因素纳入考量，如发出请求的用户、发出请求的位置以及请求的特定 API，而这些因素在您的环境中可能存在异常。GuardDuty S3 Protection 会监控 CloudTrail 收集的 S3 数据事件，并识别您环境中所有 S3 存储桶中可能存在的异常和恶意行为。

Amazon S3 访问日志

服务访问日志提供有关对存储桶做出的请求的详细记录。对于许多应用程序而言，服务器访问日志很有用。例如，访问日志信息可能在安全和访问权限审核方面很有用。有关更多信息，请参阅 使用服务器访问日志记录来记录请求。

AWS Trusted Advisor

Trusted Advisor 凝聚了从为数十万 AWS 客户提供服务中总结的最佳实践。Trusted Advisor 可检查您的 AWS 环境，然后在有可能节省开支、提高系统可用性和性能或弥补安全漏洞时为您提供建议。所有 AWS 客户均有权访问五个 Trusted Advisor 检查。使用“商业”和“企业”支持计划的客户可以查看所有 Trusted Advisor 检查。

Trusted Advisor 有以下与 Amazon S3 相关的检查：

- Amazon S3 存储桶的日志记录配置。
- 具有开放访问权限的 Amazon S3 存储桶的安全性检查。
- 未启用版本控制或已暂停版本控制的 Amazon S3 存储桶的容错检查。

有关更多信息，请参阅 AWS Support 用户指南中的 AWS Trusted Advisor。

下面的安全最佳实践也处理日志记录和监控：

- Identify and audit all your Amazon S3 buckets
- Implement monitoring using Amazon Web Services monitoring tools
- Enable AWS Config（启用 Gem）
- Enable Amazon S3 server access logging
- Use CloudTrail
- Monitor Amazon Web Services security advisories

Amazon S3 的合规性验证

作为多个 AWS 合规性计划的一部分，第三方审计员将评估 Amazon S3 的安全性和合规性，包括以下内容：

- 系统和组织控制 (SOC)
- 支付卡行业数据安全标准 (PCI DSS)
- 联邦风险与授权管理项目 (FedRAMP)
- 健康保险流通与责任法案 (HIPAA)

AWS 在合规性计划范围内的 AWS 服务中提供特定合规性计划范围内经常更新的 AWS 服务列表。

提供第三方审计报告，可使用 AWS Artifact 下载。有关更多信息，请参阅下载 AWS Artifact 中的报告。

有关 AWS 合规性计划的更多信息，请参阅 AWS 合规性计划。

您在使用 Amazon S3 时的合规性责任由您数据的敏感性、您组织的合规性目标以及适用的法律法规决定。如果您对 Amazon S3 的使用需遵守 HIPAA、PCI 或 FedRAMP 等标准，AWS 提供了以下实用资源：

- 安全性和合规性快速入门指南，介绍了架构注意事项，以及在 AWS 上部署侧重于安全性和合规性的基准环境的步骤。
- 设计符合 HIPAA 安全性和合规性要求的架构概述了公司如何使用 AWS 来帮助自己满足 HIPAA 要求。
- AWS 合规性资源提供了可能适用于您的行业和位置的多个不同业务手册和指南。
- AWS Config 可用于评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub 提供了 AWS 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。
- 使用 S3 对象锁定 可以帮助您满足金融服务监管机构（如SEC、FINRA 和 CFTC）的技术要求，这些监管机构对于某些类型的账簿和记录信息要求编写一次、多次读取 (WORM) 数据存储。
- Amazon S3 清单 可帮助您出于业务、合规性和法规要求，使用它来审核和报告对象的复制和加密状态。

Amazon S3 中的恢复能力

AWS 全球基础设施围绕区域和可用区构建。AWS 区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。这些可用区为您提供了高效的方法来设计和操作应用程序和数据库。与传统的单个数据中心基础设施或多个数据中心基础设施相比，它们具有更高的可用性、容错性和可扩展性。如果您尤其需要跨更长地理距离复制您的数据，您可以使用复制对象概述，它允许跨不同 AWS 区域中的存储桶自动异步复制对象。

每个 AWS 区域都有多个可用区。您可以跨一个区域中的多个可用区部署您的应用来获得容错性和低延迟。可用区通过快速、私密的光纤网络来互相连接，使您能够轻松构建可在可用区之间无中断地自动实现故障转移的应用程序。

有关 AWS 区域 和可用区的更多信息，请参阅 AWS 全球基础设施。

除了 AWS 全球基础设施之外，Amazon S3 还提供多种特征，以帮助支持您的数据恢复能力和备份需求。

生命周期配置

生命周期配置是一组规则，用于定义 Amazon S3 对一组对象应用的操作。利用生命周期配置规则，您可以指示 Amazon S3 将对象转换为较低成本的存储类，或者归档或删除它们。有关更多信息，请参阅 管理存储生命周期。

版本控制

版本控制是在相同的存储桶中保留对象的多个变量的方法。对于 Amazon S3 存储桶中存储的每个对象，您可以使用版本控制功能来保存、检索和还原它们的各个版本。使用版本控制能够轻松从用户意外操作和应用程序故障中恢复数据。有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

S3 对象锁定

您可以使用 S3 对象锁定通过一次写入、多次读取 (WORM) 模式存储对象。使用 S3 对象锁定时，您可以在固定的时间段内或无限期地阻止删除或覆盖对象。S3 对象锁定让您可以满足需要 WORM 存储的法规要求，或只是添加一个额外的保护层来防止对象被更改和删除。有关更多信息，请参阅使用 S3 对象锁定。

存储类

Amazon S3 提供一系列存储类，可供选择，具体取决于您的工作负载要求。S3 Standard-IA 和 S3 One Zone-IA 存储类用于大约每月访问一次且需要毫秒访问的数据。S3 Glacier Instant Retrieval 存储类专为长期归档数据而设计，您可以访问几毫秒的访问权限，大约每季度访问一次。对于不需

要立即访问的归档数据，例如备份，您可以使用 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类。有关更多信息，请参阅 使用 Amazon S3 存储类。

下面的安全最佳实践也处理弹性：

- Enable versioning
- Consider Amazon S3 cross-region replication
- Identify and audit all your Amazon S3 buckets

Amazon S3 备份的加密

如果您使用 Amazon S3 存储备份，则备份的加密取决于这些存储桶的配置。Amazon S3 提供了一种方法来设置 S3 存储桶的默认加密行为。您可以对存储桶设置默认加密，以便在存储桶中存储所有对象时对这些对象进行加密。默认加密支持存储在 AWS KMS 中的密钥（SSE-KMS）。有关更多信息，请参阅 为 Amazon S3 存储桶设置默认服务器端加密行为。

有关版本控制和对象锁定的详细信息，请参阅以下主题：在 S3 存储桶中使用版本控制 使用 S3 对象锁定

Amazon S3 中的基础设施安全性

作为一个受管理的服务，Amazon S3 受到 AWS 全局网络安全过程的保护，这些过程在 AWS Well-Architected 框架的安全支柱中进行了描述。

通过网络访问 Amazon S3 是通过 AWS 发布的 API 进行的。客户端必须支持传输层安全性（TLS）1.2。我们建议还支持 TLS 1.3。（有关此建议的更多信息，请参阅《AWS 安全博客》上的使用 TLS 1.3 加快 AWS 云连接）。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Diffie-Hellman Ephemeral (ECDHE)。另外，请求必须使用 AWS 签名 V4 或 AWS 签名 V2 进行签名，同时要求提供有效凭证。

这些 API 可以从任何网络位置调用。然而，Amazon S3 的确支持基于资源的访问策略，其中可以包含基于源 IP 地址的限制。您还可以使用 Amazon S3 存储桶策略控制从特定 Virtual Private Cloud（VPC）端点或特定 VPC 对存储桶的访问。事实上，这隔离了在 AWS 网络中仅从特定 VPC 到给定 Amazon S3 存储桶的网络访问。有关更多信息，请参阅 使用存储桶策略控制从 VPC 端点的访问。

下面的安全最佳实践也处理 Amazon S3 中的基础设施安全性：

- Consider VPC endpoints for Amazon S3 access
- Identify and audit all your Amazon S3 buckets

Amazon S3 中的配置和漏洞分析

AWS 负责处理基本安全任务，如来宾操作系统（OS）和数据库补丁、防火墙配置和灾难恢复等。这些流程已通过相应第三方审核和认证。有关更多详细信息，请参阅以下资源：

- Amazon S3 的合规性验证
- 责任共担模式
- Amazon Web Services：安全过程概述

以下安全最佳实践还处理 Amazon S3 中的配置和漏洞分析：

- Identify and audit all your Amazon S3 buckets
- Enable AWS Config（启用 Gem）

Amazon S3 的安全最佳实践

Amazon S3 提供了在您开发和实施自己的安全策略时需要考虑的大量安全特征。以下最佳实操是一般准则，并不代表完整的安全解决方案。这些最佳实践可能不适合您的环境或不满足您的环境要求，请将其视为有用的建议而不是惯例。

主题

Amazon S3 安全最佳实践

Amazon S3 的以下最佳实践可以帮助预防安全事故。

禁用访问控制列表（ACL)

S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权和禁用或启用 ACL。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对数据的访问权限。

Amazon S3 中的大多数现代使用案例不再要求使用访问控制列表（ACL）。我们建议您禁用 ACL，除非有必须单独控制每个对象的访问权限的特殊情况。要禁用 ACL 并获得存储桶中每个对象的所有权，请为 S3 对象所有权应用强制存储桶拥有者设置。禁用 ACL 时，您可以轻松通过不同的 AWS 账户 上传的对象维护存储桶。

禁用 ACL 时，数据的访问控制基于策略，如下所示：

- AWS Identity and Access Management（IAM）用户策略
- S3 存储桶策略
- 虚拟私有云（VPC）端点策略
- AWS Organizations 服务控制策略（SCP）

禁用 ACL 简化了权限管理和审计。对于新存储桶，默认情况下 ACL 处于禁用状态。您还可以对现有存储桶禁用 ACL。如果现有存储桶中已有对象，在禁用 ACL 后，对象和存储桶 ACL 将不再是访问评估过程的一部分。相反，将根据策略授予或拒绝访问权限。

在禁用 ACL 之前，请确保执行以下操作：

- 查看存储桶策略，以确保此策略涵盖了您打算在账户外授予对存储桶的访问权限的所有方式。
- 将存储桶 ACL 重置为默认（由存储桶拥有者完全控制）。

禁用 ACL 后，会出现以下行为：

- 您的存储桶仅接受未指定 ACL 的 PUT 请求或指定了存储桶拥有者完全控制的 ACL 的 PUT 请求。这些 ACL 包括 `bucket-owner-full-control` 标准 ACL 或以 XML 表示的此 ACL 的等效形式。
- 支持存储桶拥有者完全控制 ACL 的现有应用程序没有影响。
- 包含其他 ACL 的 PUT 请求（例如，向某些 AWS 账户 进行自定义授权）将失败并返回 HTTP 状态代码 `400 (Bad Request)`，和错误代码 `AccessControlListNotSupported`。

有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

确保您的 Amazon S3 存储桶使用正确的策略且不可公有访问

除非您明确要求互联网上的任何人都能读写您的 S3 存储桶，否则请确保 S3 存储桶不是公有的。以下是您可以采取的屏蔽公共访问权限的一些步骤：

- 使用 S3 屏蔽公共访问权限。借助 S3 屏蔽公共访问权限，您可以轻松地设置集中控制，以限制对 Amazon S3 资源的公共访问。无论资源是如何创建的，都会实施这些集中控制。有关更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。
- 找出允许使用通配符身份的 Amazon S3 存储桶策略，例如 `"Principal": "*"`（这实际上意味着“任何人”）。还要找出允许通配符操作 `"*"` 的策略（这实际上允许用户在 Amazon S3 存储桶中执行任何操作）。
- 类似地，找出向“任何人”或者“任何经过身份验证的 AWS 用户”提供读、写或完全访问权限的 Amazon S3 存储桶访问控制列表（ACL）。
- 使用 `ListBuckets` API 操作扫描所有 Amazon S3 存储桶。然后，使用 `GetBucketAcl`、`GetBucketWebsite` 和 `GetBucketPolicy` 确定每个存储桶是否拥有符合要求的访问控制和符合要求的配置。
- 使用 AWS Trusted Advisor 检查您的 Amazon S3 实现。
- 考虑使用 s3-bucket-public-read-prohibited 和 s3-bucket-public-write-prohibited 托管式 AWS Config 规则 来实施持续的侦测性控制。

有关更多信息，请参阅 Amazon S3 的身份和访问管理。

使用 Amazon GuardDuty 识别您的 Amazon S3 存储桶面临的潜在威胁

Amazon GuardDuty 是一项威胁检测服务，可识别您的账户、容器、工作负载以及 AWS 环境中的数据面临的潜在威胁。Amazon GuardDuty 通过使用机器学习（ML）模型以及异常和威胁检测功能，持续监控不同的数据来源，以识别环境中的潜在安全风险和恶意活动并确定其优先级。启用 GuardDuty 后，它会为包括 AWS CloudTrail 管理事件、VPC 流日志和 DNS 日志在内的基础数据来源提供威胁检测。要将威胁检测扩展到 S3 存储桶中的数据面板事件，您可以启用 GuardDuty S3 Protection 功能。此功能可检测诸如数据泄露以及通过 Tor 节点对 S3 存储桶的可疑访问之类的威胁。GuardDuty 还会在您的环境中建立常规的基准模式，并在发现潜在的异常行为时提供上下文信息，来协助您修复可能受攻击的 S3 存储桶或 AWS 凭证。有关更多信息，请参阅 GuardDuty。

实施最低权限访问

在授予权限时，您可以决定谁获得哪些 Amazon S3 资源的哪些权限。您可以对这些资源启用希望允许的特定操作。因此，我们建议您仅授予执行任务所需的权限。实施最低权限访问对于减小安全风险以及可能由错误或恶意意图造成的影响至关重要。

为实现最低权限访问，可以使用以下工具：

- Amazon S3 的策略操作 和 IAM 实体的权限边界
- Amazon S3 如何与 IAM 配合使用
- 访问控制列表 (ACL) 概述
- 服务控制策略

有关选择一个或多个前述机制时所要考虑的事项的指南，请参阅 Amazon S3 的身份和访问管理。

将 IAM 角色用于需要 Amazon S3 访问权限的应用程序和 AWS 服务

为了使在 Amazon EC2 或其他 AWS 服务上运行的应用程序能够访问 Amazon S3 资源，应用程序必须在其 AWS API 请求中包含有效的 AWS 凭证。我们建议您不要直接在应用程序或 Amazon EC2 实例中存储 AWS 凭证，这些是不会自动轮换的长期凭证，如果它们受到损害，可能会对业务产生重大影响。

而是使用 IAM 角色来管理需要访问 Amazon S3 的应用程序或服务的临时凭证。在使用角色时，您不需要将长期凭证（如用户名和密码或访问密钥）分配给 Amazon EC2 实例或 AWS 服务（例如 AWS Lambda）。角色可提供临时权限供应用程序在调用其他 AWS 资源时使用。

有关更多信息，请参阅 IAM 用户指南中的以下主题：

- IAM 角色
- 针对角色的常见情形：用户、应用程序和服务

考虑加密静态数据

您可以通过以下选项在 Amazon S3 中保护静态数据：

- 服务器端加密 – 默认情况下，所有 Amazon S3 存储桶都配置了加密，所有上传到 S3 存储桶的新对象都会自动静态加密。具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）是 Amazon S3 中每个存储桶的默认加密配置。要使用其他类型的加密，您可以指定要在 S3 PUT 请求中使用的服务器端加密类型，也可以在目标存储桶中设置默认加密配置。

  Amazon S3 也提供了以下服务器端加密选项：

  - 具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）
  - 具有 AWS Key Management Service（AWS KMS）密钥的双层服务器端加密（DSSE-KMS）
  - 具有客户提供密钥的服务器端加密（SSE-C）

  有关更多信息，请参阅 使用服务器端加密保护数据。
- 客户端加密 – 在客户端加密数据并将加密的数据上传到 Amazon S3。在这种情况下，您需要管理加密过程、加密密钥和相关的工具。如服务器端加密一样，客户端加密可以帮助减少面临的风险：通过使用存储在一个不同机制（而不是存储数据本身的机制）中的密钥来加密数据。

  Amazon S3 提供多个客户端加密选项。有关更多信息，请参阅 使用客户端加密保护数据。

实施传输中数据加密

您可以使用 HTTPS（TLS）帮助防止潜在攻击者使用中间人攻击或类似攻击来窃听或操纵网络流量。我们建议通过在 Amazon S3 存储桶策略中使用 aws:SecureTransport 条件，以只允许通过 HTTPS（TLS）的加密连接。

> **Important**
>
> 我们建议您的应用程序不要固定 Amazon S3 TLS 证书，因为 AWS 不支持固定公开信任的证书。S3 会自动续订证书，而续订可能在证书到期之前的任何时候发生。续订证书会生成新的公有-私有密钥对。如果您固定了最近使用新公钥续订的 S3 证书，则在您的应用程序使用新证书之前，您将无法连接到 S3。

也可以考虑使用 s3-bucket-ssl-requests-only 托管式 AWS Config 规则来实施持续的侦测性控制。

考虑使用 S3 对象锁定

借助 S3 对象锁定，您可以使用“一次写入，多次读取”（WORM）模式存储对象。S3 对象锁定可以帮助阻止意外或不当删除数据。例如，您可以使用 S3 对象锁定来帮助保护您的 AWS CloudTrail 日志。

有关更多信息，请参阅 使用 S3 对象锁定。

启用 S3 版本控制

S3 版本控制是在相同的存储桶中保留对象的多个版本的方法。对于 存储桶中存储的每个对象，您可以使用版本控制功能来保留、检索和还原它们的各个版本。使用版本控制能够轻松从用户意外操作和应用程序故障中恢复数据。

也可以考虑使用 s3-bucket-versioning-enabled 托管式 AWS Config 规则来实施持续的侦测性控制。

有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

考虑使用 Amazon S3 跨区域复制

虽然 Amazon S3 在默认情况下跨多个地理位置不同的可用区存储数据，但合规性要求所规定的数据存储距离可能更远。通过 S3 跨区域复制（CRR），可以在远距离 AWS 区域之间复制数据以帮助满足这些要求。CRR 允许在不同 AWS 区域 中的存储桶之间自动以异步方式复制对象。有关更多信息，请参阅 复制对象概述。

> **Note**
>
> CRR 要求源和目标 S3 存储桶都已启用版本控制。

也可以考虑使用 s3-bucket-replication-enabled 托管式 AWS Config 规则来实施持续的侦测性控制。

考虑使用 VPC 端点进行 Amazon S3 访问

Amazon S3 的 Virtual Private Cloud（VPC）端点是 VPC 内的逻辑实体，仅允许连接到 Amazon S3。VPC 端点可以帮助防止流量穿越开放的互联网。

Amazon S3 的 VPC 端点提供多种方式来控制对 Amazon S3 数据的访问：

- 通过使用 S3 存储桶策略，您可以控制允许通过特定 VPC 端点进行访问的请求、用户或组。
- 可以使用 S3 存储桶策略控制哪些 VPC 或 VPC 端点有权访问 S3 存储桶。

- 您可以使用没有 Internet 网关的 VPC 来阻止数据泄露。

有关更多信息，请参阅 使用存储桶策略控制从 VPC 端点的访问。

使用托管式 AWS 安全服务监控数据安全

多项托管式 AWS 安全服务可以帮助您识别、评测和监控 Amazon S3 数据的安全和合规风险。这些服务还可以帮助您保护数据免受这些风险的影响。这些服务包括自动检测、监控和保护功能，旨在从单个 AWS 账户的 Amazon S3 资源扩展到跨数千个账户的多个组织的资源。

有关更多信息，请参阅 使用托管式 AWS 安全服务监控数据安全。

Amazon S3 监控和审计最佳实践

Amazon S3 的以下最佳实践可以帮助检测潜在的安全弱点和事故。

识别和审计您的所有 Amazon S3 存储桶

确定您的 IT 资产是监管和安全性的一个至关重要的方面。您需要了解您所有的 Amazon S3 资源，以评估它们的安全态势并对潜在的薄弱领域采取措施。要审计您的资源，我们建议您执行以下操作：

- 使用标签编辑器确定和标记安全敏感性或审计敏感性资源，然后在您需要搜索这些资源时使用这些标签。有关更多信息，请参阅《标记 AWS 资源用户指南》中的搜索要标记的资源。
- 出于业务、合规性和法规要求，使用 S3 清单来审计和报告对象的复制和加密状态。有关更多信息，请参阅 Amazon S3 清单。
- 为您的 Amazon S3 资源创建资源组。有关更多信息，请参阅《AWS Resource Groups 用户指南》中的什么是 Resource Groups？。

使用 AWS 监控工具实施监控

监控是保持 Amazon S3 和您的 AWS 解决方案的可靠性、安全性、可用性和性能的重要部分。AWS 提供了一些可用来监控 Amazon S3 和您的其他 AWS 服务的工具和服务。例如，您可以监控 Amazon S3 的 Amazon CloudWatch 指标，特别是 `PutRequests`、`GetRequests`、`4xxErrors` 和 `DeleteRequests` 指标。有关更多信息，请参阅使用 Amazon CloudWatch 监控指标 和监控 Amazon S3。

有关另一个示例，请参阅示例：Amazon S3 存储桶活动。此示例介绍如何创建 CloudWatch 警报，当执行 Amazon S3 API 调用以 PUT 或 DELETE 存储桶策略、存储桶生命周期、存储桶复制配置时，或者 PUT 存储桶 ACL 时，会触发此警报。

启用 Amazon S3 服务器访问日志记录

服务器访问日志记录详细地记录对存储桶提出的各种请求。服务器访问日志可以在安全和访问审计方面为您提供帮助，帮您了解您的客户群和了解您的 Amazon S3 账单。有关启用服务器访问日志记录的说明，请参阅 使用服务器访问日志记录来记录请求。

也可以考虑使用 s3-bucket-logging-enabled AWS Config 托管式规则来实施持续的侦测性控制。

使用 AWS CloudTrail

AWS CloudTrail 提供用户、角色或 AWS 服务在 Amazon S3 中执行的操作的记录。您可以使用 CloudTrail 收集的信息确定以下事项：

- 向 Amazon S3 发出的请求
- 发出请求的 IP 地址
- 谁发出了请求
- 发出请求的时间
- 有关该请求的其他详细信息

例如，您可以确定影响数据访问的 PUT 操作的 CloudTrail 条目，尤其是 `PutBucketAcl`、`PutObjectAcl`、`PutBucketPolicy` 和 `PutBucketWebsite`。

当您设置 AWS 账户时，默认情况下 CloudTrail 处于启用状态。您可以在 CloudTrail 控制台中查看最近的事件。要为 Amazon S3 存储桶创建活动和事件的持续记录，您可以在 CloudTrail 控制台中创建跟踪。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的记录数据事件。

创建跟踪时，可以配置 CloudTrail 以记录数据事件。数据事件是在资源上或在资源内执行的资源操作的记录。在 Amazon S3 中，数据事件记录单个存储桶的对象级 API 活动。CloudTrail 支持 Amazon S3 对象级 API 操作的子集，例如 `GetObject`、`DeleteObject` 和 `PutObject`。有关 CloudTrail 如何与 Amazon S3 配合使用的更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon S3 API 调用。在 Amazon S3 控制台中，您还可以将 S3 存储桶配置为 为 S3 存储桶和对象启用 CloudTrail 事件日志记录。

AWS Config 提供了托管规则（`cloudtrail-s3-dataevents-enabled`），您可以使用该规则来确认至少有一个 CloudTrail 跟踪正在记录 S3 存储桶的数据事件。有关更多信息，请参阅 AWS Config 开发人员指南中的 cloudtrail-s3-dataevents-enabled。

Enable AWS Config（启用 Gem）

此主题中列出的几个最佳实践建议创建 AWS Config 规则。AWS Config 有助于评测、审计和评价您的 AWS 资源的配置。AWS Config 监控资源配置，以便您能够针对需要的安全配置评估所记录的配置。利用 AWS Config，您可以：

- 查看 AWS 资源之间的配置和关系的更改
- 调查详细的资源配置历史记录
- 根据内部指南中指定的配置确定总体合规性

使用 AWS Config 可帮助您简化合规性审核、安全性分析、变更管理和操作故障排除。有关更多信息，请参阅 AWS Config 开发人员指南中的使用控制台设置 AWS Config。当指定要记录的资源类型时，确保您包括了 Amazon S3 资源。

> **Important**
>
> 在评估 Amazon S3 资源时，AWS Config 托管式规则仅支持通用存储桶。AWS Config 不记录目录存储桶的配置更改。有关更多信息，请参阅《AWS Config 开发人员指南》中的 AWS Config 托管式规则和 AWS Config 托管式规则列表。

有关如何使用 AWS Config 的示例，请参阅《AWS 安全博客》上的如何使用 AWS Config 来监控和响应允许公共访问的 Amazon S3 存储桶。

使用 Amazon Macie 发现敏感数据

Amazon Macie 是一项安全服务，旨在使用机器学习和模式匹配来发现敏感数据。Macie 提供对数据安全风险的可见性，并实现针对这些风险的自动防护。借助 Macie，您可以自动发现和报告 Amazon S3 数据资产中的敏感数据，以更好地了解组织在 S3 中存储的数据。

要使用 Macie 检测敏感数据，您可以使用内置标准和技术，这些标准和技术旨在检测许多国家和地区的大量且不断增长的敏感数据类型列表。这些敏感数据类型包括多种类型的个人身份信息（PII）、财务数据和凭证数据。您还可以使用自己定义的自定义标准：旨在定义要匹配的文本模式的正则表达式，以及（可选）用于优化结果的字符序列和邻近规则。

如果 Macie 在 S3 对象中检测到敏感数据，Macie 会生成安全调查发现来通知您。此调查发现提供有关受影响的对象、Macie 发现的敏感数据的类型和出现次数的信息，以及其他详细信息，以帮助您调查受影响的 S3 存储桶和对象。有关更多信息，请参阅《Amazon Macie 用户指南》。

使用 S3 Storage Lens 存储统计管理工具

S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。S3 Storage Lens 存储统计管理工具还分析指标以提供上下文建议，您可以使用这些建议来优化存储成本并应用最佳实践来保护数据。

通过 S3 Storage Lens 存储统计管理工具，您可以使用指标生成摘要见解，例如，了解整个组织中有多少存储空间，或增长最快的存储桶和前缀是哪些。您还可以使用 S3 Storage Lens 存储统计管理工具指标来识别成本优化机会，实施数据保护和访问管理最佳实践，并提高应用程序工作负载的性能。

例如，您可以识别没有 S3 生命周期规则的存储桶，以中止超过 7 天的未完成分段上传。您还可以识别未遵循数据保护最佳实践（例如使用 S3 复制或 S3 版本控制）的存储桶。有关更多信息，请参阅了解 Amazon S3 Storage Lens。

监控 AWS 安全公告

我们建议您经常为您的 AWS 账户检查在 Trusted Advisor 中发布的安全公告。尤其要关注有关具有“开放访问权限”的 Amazon S3 存储桶的警告。您可以使用 describe-trusted-advisor-checks 通过编程方式来实现这一点。

此外，积极地监控向您的每一个 AWS 账户注册的原始电子邮件地址。AWS 将使用该电子邮件地址就可能影响您的紧急安全事件与您联系。

具有广泛影响的 AWS 操作性问题将在 AWS Health Dashboard 服务运行状况上发布。操作性问题也通过 AWS Health Dashboard 发布到各个账户。有关更多信息，请参阅 AWS Health 文档。

使用托管式 AWS 安全服务监控数据安全

多项托管式 AWS 安全服务可以帮助您识别、评测和监控 Amazon S3 数据的安全和合规风险。这些服务还可以帮助您保护数据免受这些风险的影响。这些服务包括自动检测、监控和保护功能，旨在从单个 AWS 账户的 Amazon S3 资源扩展到跨数千个 AWS 账户的多个组织的资源。

AWS 检测和响应服务可以帮助您识别潜在的安全配置错误、威胁或意外行为，以便您可以快速响应环境中潜在未经授权或恶意的活动。AWS 数据保护服务可以帮助您监控和保护您的数据、账户和工作负载，使其免受未经授权的访问。这些服务还有助于您发现 Amazon S3 数据资产中的敏感数据 [如个人身份信息（PII）]。

为帮助您识别和评估数据安全与合规性风险，托管式 AWS 安全服务会生成调查发现，以告知您与 Amazon S3 数据有关的潜在安全事件或问题。调查发现提供相关详细信息，您可以使用这些信息，根据事件/响应工作流和策略对这些风险进行调查、评测并采取措施。您可以使用每项服务直接访问调查发现数据。还可以将数据发送到其他应用程序、服务和系统，例如安全事故和事件管理系统（SIEM）。

要监控 Amazon S3 数据的安全性，请考虑使用以下这些托管式 AWS 安全服务。

Amazon GuardDuty

Amazon GuardDuty 是一项威胁检测服务，可持续监控您的 AWS 账户和工作负载中是否存在恶意活动，并提供详细的安全调查发现以供查看和补救。

Amazon GuardDuty 提供了两种防护计划，您可以使用它们来检测 S3 存储桶中的潜在威胁。

GuardDuty S3 防护

- 启用 S3 防护时，GuardDuty 将监控 Amazon S3 资源的 AWS CloudTrail 管理和数据事件。然后，GuardDuty 会分析这些事件，例如潜在的数据泄露和数据破坏。为了为分析提供信息并识别潜在的安全风险，GuardDuty 使用威胁情报源和机器学习。
- GuardDuty 可以监控 Amazon S3 资源的不同类型的活动。例如，Amazon S3 的 CloudTrail 管理事件包括存储桶级操作，例如 `ListBuckets`、`DeleteBucket` 和 `PutBucketReplication`；Amazon S3 的 CloudTrail 数据事件包括对象级操作，例如 `GetObject`、`ListObjects` 和 `PutObject`。如果 GuardDuty 检测到异常或潜在恶意的活动，它会生成调查发现来通知您。
- 有关更多信息，请参阅《Amazon GuardDuty 用户指南》中的 GuardDuty S3 Protection。

S3 恶意软件防护

- 启用此防护计划可扫描新上传到所选 S3 存储桶的对象，以查找潜在的恶意软件。也可以将恶意软件扫描的范围限制为特定的对象前缀。

- 这项特性提供的功能包括使用 Amazon EventBridge 自动监控扫描状态、使用 Amazon CloudWatch 评测扫描指标以及可选标记恶意 S3 对象等。
- GuardDuty 提供了无需启用 GuardDuty 服务即可启用 S3 恶意软件防护的选项。有关更多信息，请参阅《Amazon GuardDuty 用户指南》中的 GuardDuty Malware Protection for S3。

Amazon Detective

Amazon Detective 简化了调查流程，并帮助您进行更快、更有效的安全调查。Detective 提供预构建的数据聚合、摘要和上下文，可以帮助您分析和评测可能的安全问题的性质和程度。

Detective 会自动提取基于时间的事件，例如来自 AWS CloudTrail 的 API 调用和您的 AWS 资源的 Amazon VPC 流日志。它还采集 Amazon GuardDuty 生成的调查发现。然后，Detective 使用机器学习、统计分析和图形理论生成可视化效果，以帮助您更快地进行有效的安全调查。

这些可视化效果提供了统一的交互式视图，可供您了解资源行为及此类行为随时间推移的相互作用。您可以浏览此行为图以检查潜在的恶意操作，例如登录尝试失败或可疑的 API 调用。您还可以查看这些操作如何影响资源（如 S3 存储桶和对象）。

有关更多信息，请参阅《Amazon Detective 管理指南》。

IAM Access Analyzer

AWS Identity and Access Management Access Analyzer（IAM Access Analyzer）可帮助您识别与外部实体共享的资源。还可以使用 IAM Access Analyzer 根据策略语法和最佳实践验证 IAM 策略，并根据 AWS CloudTrail 日志中的访问活动生成 IAM 策略。

IAM Access Analyzer 使用基于逻辑的推理来分析 AWS 环境中的资源策略，例如存储桶策略。借助适用于 S3 的 IAM Access Analyzer，当 S3 存储桶配置为允许互联网上的任何人或其他 AWS 账户（包括组织外部的账户）访问时，将向您发出提醒。例如，适用于 S3 的 IAM Access Analyzer 可能会报告存储桶具有通过存储桶访问控制列表（ACL）、存储桶策略、多区域接入点策略或接入点策略提供的读取或写入访问权限。您会收到每个公共存储桶或共享存储桶的调查发现，其中指出了公共或共享访问的来源和级别。有了这些调查发现，您就可以立即采取精确的纠正措施，将存储桶访问权限恢复为您期望的设置。

有关更多信息，请参阅 使用适用于 S3 的 IAM Access Analyzer 查看存储桶访问权限。

Amazon Macie

Amazon Macie 是一项数据安全服务，该服务使用机器学习和模式匹配来发现敏感数据，提供对数据安全风险的可见性，并实现针对这些风险的自动防护。

借助 Macie，您可以自动发现和报告 S3 存储桶中的敏感数据，以更好地了解组织在 Amazon S3 中存储的数据。要检测敏感数据，您可以使用 Macie 提供的内置标准和技术、您定义的自定义标准或

两者的组合。如果 Macie 在 S3 对象中检测到敏感数据，Macie 会生成调查发现来通知您。此调查发现提供有关受影响的存储桶和对象、Macie 发现的敏感数据的类型和出现次数的信息，以及其他有助于调查的详细信息。

Macie 还提供统计数据和其他数据，让您深入了解 Amazon S3 数据的安全状况，并且自动评估和监控您的 S3 存储桶以实现安全性和访问控制。如果 Macie 检测到潜在的数据安全性或隐私问题（例如存储桶变为可供公共访问），Macie 会生成调查发现，供您查看并在必要时进行补救。

有关更多信息，请参阅《Amazon Macie 用户指南》。

AWS Security Hub

AWS Security Hub 是一项安全态势管理服务，该服务可执行安全最佳实践检查，将来自多个来源的警报和调查发现聚合为单一格式，并支持自动补救。

Security Hub 从集成的 AWS Partner Network 安全解决方案和 AWS 服务（包括 Amazon Detective、Amazon GuardDuty、IAM Access Analyzer 和 Amazon Macie）收集并提供安全调查发现数据。Security Hub 还可以根据 AWS 最佳实践和支持的行业标准运行持续的自动化安全检查，从而生成自己的调查发现。

然后，Security Hub 关联并整合各提供方的调查发现，以帮助确定调查发现的优先级并处理最重要的调查发现。它还为自定义操作提供支持，您可以使用自定义操作来为特定类别的调查发现调用响应或补救操作。

使用 Security Hub，您可以评测 Amazon S3 资源的安全性和合规性状态，也可以将其作为对组织在单个 AWS 区域和跨多个区域的安全状况进行更广泛分析的一部分。这包括分析安全趋势和确定优先级最高的安全问题。您还可以聚合来自多个 AWS 区域的调查发现，并监控和处理来自单个区域的聚合调查发现数据。

有关更多信息，请参阅《AWS Security Hub 用户指南》中的 Amazon Simple Storage Service 控制。

管理您的 Amazon S3 存储

在 Amazon S3 中创建存储桶和上传对象后，您可以使用版本控制、存储类、对象锁定、批量操作、复制、标签等功能来管理对象存储。以下各部分提供了有关 Amazon S3 中可用的存储管理功能和特性的详细信息。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅什么是 S3 Express One Zone？和目录桶。

主题

在 S3 存储桶中使用版本控制

Amazon S3 中的版本控制是在相同的存储桶中保留对象的多个变量的方法。对于存储桶中存储的每个对象，您可以使用 S3 版本控制功能来保留、检索和还原它们的各个版本。使用版本控制能够更加轻松

地从用户意外操作和应用程序故障中恢复数据。为存储桶启用版本控制后，如果 Amazon S3 同时收到针对同一对象的多个写入请求，它会存储所有对象。

启用了版本控制的存储桶可以协助您恢复因意外删除或覆盖操作而失去的对象。例如，如果您删除对象，Amazon S3 会插入删除标记，而不是永久删除该对象。删除标记将成为当前对象版本。如果覆盖对象，则会导致存储桶中出现新的对象版本。您始终可以恢复以前的版本。有关更多信息，请参阅 从启用了版本控制的存储桶中删除对象版本。

默认情况下，S3 版本控制在存储桶上处于禁用状态，您必须明确启用它。有关更多信息，请参阅 在存储桶上启用版本控制。

> **Note**
>
> - SOAP API 不支持 S3 版本控制。HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。
> - 对于存储和传输的每个对象版本，都适用正常 Amazon S3 费率。对象的每个版本都是完整的对象；它并非只是与上一版本有所不同。因此，如果您存储了三个版本的对象，则会收取您三个对象的费用。

不受版本控制、启用了版本控制和已暂停版本控制的存储桶

存储桶可能处于以下三种状态之一：

- 不受版本控制（默认）
- 已启用版本控制
- 已暂停版本控制

您可以在存储桶级别启用和暂停版本控制。一旦您对存储桶启用了版本控制，它将无法返回到不受版本控制状态。但是，您可以在该存储桶上暂停版本控制。

版本控制状态将应用到该存储桶中的所有（不是某些）对象。当您在存储桶中启用版本控制功能时，所有新对象都将受版本控制，并为其指定唯一的版本 ID。此后，在启用版本控制功能时，存储桶中已存在的对象将始终受版本控制，并在以后的请求修改时为其提供唯一的版本 ID。请注意以下几点：

- 在您设置版本控制状态之前存储在存储桶中的对象的版本 ID 为 `null`。启用版本控制时，存储桶中的现有对象不会更改。更改的是 Amazon S3 在以后的请求中处理这些对象的方式。有关更多信息，请参阅 使用启用版本控制的存储桶中的对象。

- 存储桶拥有者（或任何具有适当权限的用户）可以暂停版本控制以停止累积对象版本。暂停版本控制时，存储桶中的现有对象不会更改。更改的是 Amazon S3 在以后的请求中处理对象的方式。有关更多信息，请参阅 使用已暂停版本控制的存储桶中的对象。

将 S3 版本控制与 S3 生命周期结合使用

要自定义您的数据保留方法和控制存储成本，请将对象版本控制与 S3 生命周期结合使用。有关更多信息，请参阅 管理存储生命周期。有关使用 AWS Management Console、AWS CLI、AWS SDK 或 REST API 创建 S3 生命周期配置的信息，请参阅在存储桶上设置生命周期配置。

> **Important**
>
> 如果您在不受版本控制的存储桶中具有对象到期生命周期配置，并且希望在启用版本控制时保持相同的永久删除行为，则必须添加非当前版本到期策略。非当前版本到期生命周期配置将管理在启用版本控制的存储桶中删除非当前对象版本的行为。（启用版本控制的存储桶会维护一个当前对象版本，以及零个或更多非当前对象版本。）有关更多信息，请参阅 在存储桶上设置生命周期配置。

有关使用 S3 版本控制的信息，请参阅以下主题。

主题

- S3 版本控制的工作原理
- 在存储桶上启用版本控制
- 配置 MFA 删除
- 使用启用版本控制的存储桶中的对象
- 使用已暂停版本控制的存储桶中的对象

S3 版本控制的工作原理

您可以使用 S3 版本控制将对象的多个版本保存在一个存储桶中，以便您能够还原意外删除或覆盖的对象。例如，如果您将 S3 版本控制应用于存储桶，则会发生以下变化：

- 如果删除对象（而不是永久移除此对象），则 Amazon S3 会插入删除标记，它将成为当前对象版本。然后，您可以恢复以前的版本。有关更多信息，请参阅 从启用了版本控制的存储桶中删除对象版本。

- 如果覆盖对象，则 Amazon S3 会在存储桶中添加新的对象版本。之前的版本保留在存储桶中，成为非当前版本。您可以还原以前的版本。

> **Note**
>
> 对于存储和传输的每个对象版本，都适用正常 Amazon S3 费率。对象的每个版本都是完整的对象；它与以前的版本没有什么区别。因此，如果您存储了三个版本的对象，则会收取您三个对象的费用。

您创建的每个 S3 存储桶都具有关联的 versioning 子资源。（有关更多信息，请参阅 存储桶配置选项。）默认情况下，您的存储桶不受版本控制，且对子资源进行版本控制会存储空的版本控制配置，如下所示。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

要启用版本控制，您可以向 Amazon S3 发送请求，在请求中指定包含 `Enabled` 状态的版本控制配置。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

要暂停版本控制，请将状态值设置为 `Suspended`。

> **Note**
>
> 当您首次对存储桶启用版本控制时，可能只需很短的时间即可完全传播更改。我们建议您在启用版本控制后等待 15 分钟，然后再对存储桶中的对象发出写入操作（`PUT` 或 `DELETE`）。

存储桶拥有者和所有获得授权的 AWS Identity and Access Management（IAM）用户都可以启用版本控制。创建存储桶的存储桶拥有者为 AWS 账户。有关权限的更多信息，请参阅Amazon S3 的身份和访问管理。

有关使用 AWS Management Console、AWS Command Line Interface（AWS CLI）或 REST API 启用和禁用 S3 版本控制的更多信息，请参阅the section called “在存储桶上启用版本控制”。

主题

版本 ID

如果您为存储桶启用版本控制，Amazon S3 会自动为要存储的对象生成唯一版本 ID。例如，在一个存储桶中，您可以拥有两个具有相同键（对象名称）的对象，但版本 ID 不同，例如 photo.gif（版本 111111）和 photo.gif（版本 121212）。
图中描述了一个启用了版本控制的存储桶，该存储桶有两个具有相同键但版本 ID 不同的对象。

无论 S3 版本控制是否启用，每个对象都有一个版本 ID。如果未启用 S3 版本控制，Amazon S3 将版本 ID 的值设置为 null。如果您启用 S3 版本控制，Amazon S3 会为对象分配版本 ID 值。此值将该对象与同一个键的其他版本区分开来。

在现有存储桶上启用 S3 版本控制时，已存储在存储桶中的对象将保持不变。版本 ID（null）、内容和权限保持不变。启用 S3 版本控制后，添加到存储桶的每个对象都会获得一个版本 ID，该 ID 将此版本与同一个键的其他版本区分开来。

Amazon S3 仅生成版本 ID，不能编辑它们。版本 ID 是 Unicode、UTF-8 编码、URL 就绪、不透明的字符串，长度不超过 1,024 字节。以下是示例：

```
3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
```

> **Note**
>
> 为简单起见，本主题中的其他示例使用更短的 ID。

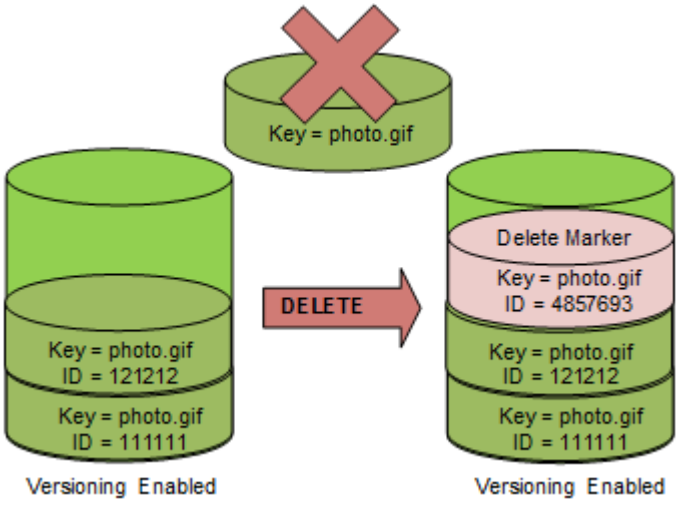
版本控制工作流程

当您在启用版本控制的存储桶中通过 PUT 放入对象时，不会覆盖非当前版本。如下图所示，当将 photo.gif 的新版本 PUT 到一个已经包含同名对象的存储桶中时，会发生以下行为：

- 原始对象（ID = 111111）保留在存储桶中。
- Amazon S3 生成一个新的版本 ID（121212），并将这个较新版本的对象添加到存储桶中。

使用此功能，如果对象被意外覆盖或删除，则可以检索对象的先前版本。

当 DELETE 对象时，所有版本都将保留在存储桶中，而 Amazon S3 将插入删除标记，如下图所示。



删除标记将成为对象的当前版本。默认情况下，GET 请求将检索最新存储的版本。在当前版本为删除标记时，执行 GET Object 请求将返回 404 Not Found 错误，如下图所示。

但是，您可以通过指定对象版本 ID，通过 GET 获取非当前版本的对象。在下图中，GET 特定对象版本 111111。即使该对象版本不是当前版本，Amazon S3 也会返回它。

有关更多信息，请参阅 从启用了版本控制的存储桶中检索对象版本。

您可以通过指定要删除的版本来永久删除对象。只有 Amazon S3 存储桶的拥有者或者授权的 IAM 用户才能永久删除某个版本。如果您的 DELETE 操作指定了 versionId，则会永久删除该对象版本，Amazon S3 不会插入删除标记。

您可以通过配置存储桶来启用多重身份验证（MFA）删除，从而提升安全性。对存储桶启用 MFA 删除时，存储桶拥有者必须在任何请求中包含两种形式的身份验证，以删除版本或更改存储桶的版本控制状态。有关更多信息，请参阅 配置 MFA 删除。

何时为对象创建新版本？

仅当您 PUT 新对象时，才会创建对象的新版本。请注意，某些操作（如 CopyObject）通过实现 PUT 操作来发挥作用。

某些修改当前对象的操作不会创建新版本，因为它们没有 PUT 新对象。这包括更改对象上的标签等操作。

> **Important**
>
> 如果您注意到，在对存储桶启用 S3 版本控制后，Amazon S3 对该存储桶的 PUT 或 DELETE 对象请求收到的 HTTP 503（服务不可用）响应数量显著增加，那么该存储桶中可能有一个或多个对象有数以百万计的版本。有关更多信息，请参阅故障排除的“S3 版本控制”一节。

在存储桶上启用版本控制

您可以使用 S3 版本控制在一个存储桶中保留对象的多个版本。本部分提供了如何使用控制台、REST API、AWS SDK 和 AWS Command Line Interface（AWS CLI）在存储桶上启用版本控制的示例。

> **Note**
>
> 如果您首次对桶启用版本控制，则可能需要长达 15 分钟的时间才能完全传播更改。我们建议您在启用版本控制后等待 15 分钟，然后再对存储桶中的对象发出写入操作（PUT 或 DELETE）。在此转换完成之前发出的写入操作可能会应用于未进行版本控制的对象。

有关 S3 版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。有关使用已启用版本控制的存储桶中的对象的信息，请参阅 使用启用版本控制的存储桶中的对象。

要了解有关如何使用 S3 版本控制保护数据的更多信息，请参阅教程：使用 S3 版本控制、S3 对象锁定和 S3 复制保护 Amazon S3 上的数据免遭意外删除或应用程序错误。

您创建的每个 S3 存储桶都具有关联的 versioning 子资源。（有关更多信息，请参阅 存储桶配置选项。）默认情况下，您的存储桶不受版本控制，且对子资源进行版本控制会存储空的版本控制配置，如下所示。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

要启用版本控制，您可以向 Amazon S3 发送一个请求，在请求中指定包含状态的版本控制配置。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

要暂停版本控制，请将状态值设置为 Suspended。

存储桶拥有者和所有授权用户都可以启用版本控制。存储桶拥有者是创建存储存储桶的 AWS 账户（根账户）。有关许可的更多信息，请参阅Amazon S3 的身份和访问管理。

以下部分提供了有关使用控制台、AWS CLI 和 AWS SDK 启用 S3 版本控制的更多详细信息。

使用 S3 控制台

请按照以下步骤使用 AWS Management Console 在 S3 存储桶上启用版本控制。

在 S3 存储桶上启用或禁用版本控制

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择要为其启用版本控制的存储桶的名称。
3. 请选择属性。
4. 在存储桶版本控制下，请选择编辑。
5. 请选择 Suspend (暂停) 或 Enable (启用)，然后选择 Save changes (保存更改)。

> **Note**
>
> 您可以将 AWS Multi-Factor Authentication (MFA) 与版本控制结合使用。将 MFA 与版本控制结合使用时，您必须提供 AWS 账户的访问密钥和账户 MFA 设备中的有效代码，才能永久删除对象版本或暂停或重新激活版本控制。
> 要将 MFA 与版本控制结合使用，请启用 `MFA Delete`。但是，您无法使用 AWS Management Console 启用 `MFA Delete`。您必须使用 AWS Command Line Interface (AWS CLI) 或 API。有关更多信息，请参阅 配置 MFA 删除。

使用 AWS CLI

以下示例在 S3 存储桶上启用版本控制。

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled
```

以下示例在存储桶上启用 S3 版本控制和多重身份验证（MFA）删除。

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

> **Note**
>
> 使用 MFA 删除需要获得批准的物理或虚拟身份验证设备。有关在 Amazon S3 中使用 MFA 删除的更多信息，请参阅配置 MFA 删除。

有关使用 AWS CLI 启用版本控制的更多信息，请参阅《AWS CLI 命令参考》中的 put-bucket-versioning。

使用 AWS SDK

以下示例在存储桶上启用版本控制，然后使用 AWS SDK for Java 和 AWS SDK for .NET 检索版本控制状态。有关使用其他 AWS SDK 的信息，请参阅 AWS 开发人员中心。

.NET

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using System;
using Amazon.S3;
using Amazon.S3.Model;

namespace s3.amazon.com.docsamples
{
    class BucketVersioningConfiguration
    {
        static string bucketName = "*** bucket name ***";

        public static void Main(string[] args)
        {
            using (var client = new AmazonS3Client(Amazon.RegionEndpoint.USEast1))
            {
                try
                {
                    EnableVersioningOnBucket(client);
```

```
                    string bucketVersioningStatus =
RetrieveBucketVersioningConfiguration(client);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    if (amazonS3Exception.ErrorCode != null &&
                        (amazonS3Exception.ErrorCode.Equals("InvalidAccessKeyId")
                        ||
                        amazonS3Exception.ErrorCode.Equals("InvalidSecurity")))
                    {
                        Console.WriteLine("Check the provided AWS Credentials.");
                        Console.WriteLine(
                        "To sign up for service, go to http://aws.amazon.com/s3");
                    }
                    else
                    {
                        Console.WriteLine(
                         "Error occurred. Message:'{0}' when listing objects",
                         amazonS3Exception.Message);
                    }
                }
            }

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }

        static void EnableVersioningOnBucket(IAmazonS3 client)
        {

                PutBucketVersioningRequest request = new PutBucketVersioningRequest
                {
                    BucketName = bucketName,
                    VersioningConfig = new S3BucketVersioningConfig
                    {
                        Status = VersionStatus.Enabled
                    }
                };

                PutBucketVersioningResponse response =
client.PutBucketVersioning(request);
        }
```

```
        static string RetrieveBucketVersioningConfiguration(IAmazonS3 client)
        {
                GetBucketVersioningRequest request = new GetBucketVersioningRequest
                {
                    BucketName = bucketName
                };

                GetBucketVersioningResponse response =
 client.GetBucketVersioning(request);
                return response.VersioningConfig.Status;
            }
    }
}
```

Java

有关如何创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;

public class BucketVersioningConfigurationExample {
    public static String bucketName = "*** bucket name ***";
    public static AmazonS3Client s3Client;

    public static void main(String[] args) throws IOException {
        s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
        s3Client.setRegion(Region.getRegion(Regions.US_EAST_1));
        try {

            // 1. Enable versioning on the bucket.
         BucketVersioningConfiguration configuration =
           new BucketVersioningConfiguration().withStatus("Enabled");

   SetBucketVersioningConfigurationRequest setBucketVersioningConfigurationRequest
 =
     new SetBucketVersioningConfigurationRequest(bucketName,configuration);
```

```
 s3Client.setBucketVersioningConfiguration(setBucketVersioningConfigurationRequest);

   // 2. Get bucket versioning configuration information.
   BucketVersioningConfiguration conf =
 s3Client.getBucketVersioningConfiguration(bucketName);
    System.out.println("bucket versioning configuration status:    " +
 conf.getStatus());

        } catch (AmazonS3Exception amazonS3Exception) {
            System.out.format("An Amazon S3 error occurred. Exception: %s",
 amazonS3Exception.toString());
        } catch (Exception ex) {
            System.out.format("Exception: %s", ex.toString());
        }
    }
}
```

Python

以下 Python 代码示例创建一个 Amazon S3 存储桶，启用它进行版本控制，并配置一个在 7 天后使非当前对象版本过期的生命周期。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
```

```
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
```

```
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

配置 MFA 删除

在 Amazon S3 存储桶中使用 S3 版本控制时，您可以选择通过将存储桶配置为启用 MFA（多重身份验证）删除来添加另一层安全保护。执行此操作时，存储桶拥有者必须在任何请求中包含两种形式的身份验证，以删除版本或更改存储桶的版本控制状态。

MFA 删除要求对以下任一操作进行额外身份验证：

- 更改存储桶的版本控制状态
- 永久删除对象版本

MFA 删除需要结合使用两种形式的身份验证：

- 您的安全凭证
- 在已批准的身份验证设备上显示的有效序列号、空格和 6 位代码的组合

MFA 删除因此可在某些情况下（例如在安全凭证被泄露时）提供更高的安全性。MFA 删除要求启动删除操作的用户证明其实际拥有具有 MFA 代码的 MFA 设备，并为删除操作增加额外的摩擦和安全性，从而帮助防止存储桶被意外删除。

要识别已启用了 MFA 删除的存储桶，您可以使用 Amazon S3 Storage Lens 存储统计管理工具指标。S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。有关更多信息，请参阅使用 S3 Storage Lens 存储统计管理工具访问存储活动和使用情况。有关指标的完整列表，请参阅 S3 Storage Lens 存储统计管理工具指标词汇表。

存储桶拥有者、创建存储存储桶的 AWS 账户（根账户）以及所有授权用户都可以启用版本控制。但是，只有存储桶拥有者（根账户）才能启用 MFA 删除。有关更多信息，请参阅 AWS 安全博客中的使用 MFA 保护对 AWS 的访问。

> **Note**
>
> 要将 MFA 删除与版本控制结合使用，请启用 `MFA Delete`。但是，您无法使用 AWS Management Console 启用 `MFA Delete`。您必须使用 AWS Command Line Interface (AWS CLI) 或 API。
>
> 有关将 MFA 删除与版本控制结合使用的示例，请参阅主题 在存储桶上启用版本控制 中的示例部分。
>
> 不能将 MFA 删除与生命周期配置一起使用。有关生命周期配置以及如何与其他配置交互的详细信息，请参阅 生命周期和其他存储桶配置。

要启用或禁用 MFA 删除，请使用对存储桶配置版本控制时所用的相同 API。Amazon S3 将 MFA 删除配置存储在用于存储桶版本控制状态的相同 versioning 子资源中。

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>VersioningState</Status>
  <MfaDelete>MfaDeleteState</MfaDelete>
</VersioningConfiguration>
```

要使用 MFA 删除，您可以使用硬件或虚拟 MFA 设备来生成身份验证代码。以下示例显示了在硬件设备上显示的生成的身份验证代码。

MFA 删除和受 MFA 保护的 API 访问的功能旨在提供对不同方案的保护。您可以在存储存储桶上配置 MFA 删除，以协助确保存储桶中的数据不会被意外删除。受 MFA 保护的 API 访问用于在访问敏感的 Amazon S3 资源时，强制执行其他身份验证因素（MFA 代码）。您可以要求使用 MFA 创建的临时证书来完成针对这些 Amazon S3 资源的任何操作。有关示例，请参阅需要 MFA。

有关如何购买和激活身份验证设备的更多信息，请参阅多重身份验证。

启用 S3 版本控制并配置 MFA 删除

使用 AWS CLI

以下示例在存储桶上启用 S3 版本控制和多重身份验证（MFA）删除。

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "SERIAL 123456"
```

使用 REST API

有关使用 Amazon S3 REST API 指定 MFA 删除的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketVersioning。

使用启用版本控制的存储桶中的对象

在您设置版本控制状态之前存储在 Amazon S3 存储桶中的对象的版本 ID 为 null。启用版本控制时，存储桶中的现有对象不会更改。更改的是 Amazon S3 在以后的请求中处理这些对象的方式。

转换对象版本

可为具有明确定义的生命周期的对象定义生命周期配置规则，以便在对象生命周期的特定时间将对象版本转换为 S3 Glacier Flexible Retrieval 存储类。有关更多信息，请参阅 管理存储生命周期。

此部分中的主题说明在启用了版本控制的存储桶中进行的各种对象操作。有关版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。

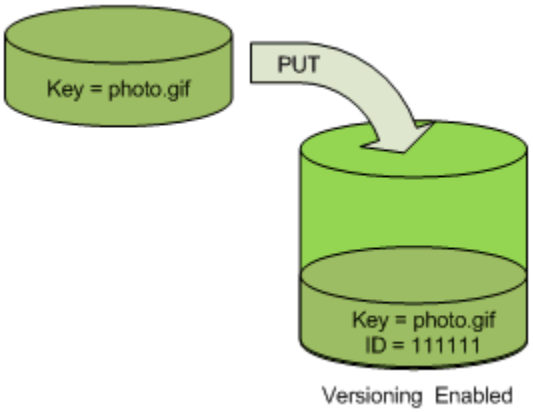
主题

- 将对象添加到已启用版本控制的存储桶
- 列出启用版本控制的存储桶中的对象
- 从启用了版本控制的存储桶中检索对象版本
- 从启用了版本控制的存储桶中删除对象版本
- 配置受版本控制的对象权限

将对象添加到已启用版本控制的存储桶

您在存储桶上启用了版本控制后，Amazon S3 会自动将唯一的版本 ID 添加到存储在该存储桶中的每个对象（使用 PUT、POST 或 CopyObject）。

下图显示当将对象添加到已启用版本控制的存储桶时，Amazon S3 会向其添加唯一的版本 ID。



> **Note**
>
> Amazon S3 分配的版本 ID 值是 URL 安全的（可以包含在 URI 中）。

有关版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。您可以使用控制台、AWS SDK 和 REST API 将对象版本添加到启用版本控制的存储桶中。

使用 控制台

有关说明，请参阅上传对象。

使用 AWS SDK

有关使用适用于 Java、.NET 和 PHP 的 AWS SDK 上传对象的示例，请参阅 上传对象。在无版本控制和启用版本控制的存储桶中上传对象的示例是相同的，只是对于启用版本控制的存储桶，Amazon S3 会分配版本号。否则，版本号为空。

有关使用其他 AWS SDK 的信息，请参阅 AWS 开发人员中心。

使用 REST API

要将对象添加到已启用版本控制的存储桶

1. 使用 `PutBucketVersioning` 请求在存储桶上启用版本控制。

   有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketVersioning。
2. 发送 `PUT`、`POST` 或 `CopyObject` 请求，以在存储桶中存储对象。

当您将对象添加到已启用版本控制的存储桶时，Amazon S3 将在 `x-amz-version-id` 响应标头中返回该对象的版本 ID，如下面的示例所示。

```
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY
```

列出启用版本控制的存储桶中的对象

此部分提供从启用版本控制的存储桶列出对象版本的示例。Amazon S3 将对象版本信息存储在与存储桶关联的 versions 子资源中。有关更多信息，请参阅 存储桶配置选项。要列出已启用版本控制的存储桶中的对象，您需要 `ListBucketVersions` 权限。

使用 S3 控制台

请按照以下步骤使用 Amazon S3 控制台查看对象的不同版本。

查看对象的多个版本

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择包含对象的存储桶的名称。
3. 要查看存储桶中对象的版本列表，请选择显示版本开关。

   控制台会显示每个对象版本的唯一版本 ID、对象版本的创建日期和时间以及其他属性。（在您设置版本控制状态之前存储在存储桶中的对象具有版本 ID null。）

   要列出没有版本的对象，请选择 List versions (列出版本) 开关。

您也可以在控制台中的对象概述面板中查看、下载和删除对象版本。有关更多信息，请参阅 在 Amazon S3 控制台中查看对象概述。

> **Note**
>
> 要访问早于 300 个版本的对象版本，必须使用 AWS CLI 或对象的 URL。

> **Important**
>
> 仅当对象已作为最新（当前）版本删除时，您才能取消删除它。您无法取消删除已删除对象的早期版本。有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

使用 AWS SDK

此部分中的示例说明了如何从启用了版本控制的存储桶中检索对象列表。每个请求最多返回 1000 个版本，除非您指定一个较小的数字。如果存储桶包含的版本数超过此限制，请发送一系列请求来检索所有版本的列表。以“页面”形式返回结果的过程称为分页。

为了说明分页的工作原理，该示例将每个响应限制为两个对象版本。在检索第一页结果后，每个示例将检查以确定版本列表是否已被截断。如果是，示例将继续检索页面，直至检索到所有版本。

> **Note**
>
> 以下示例还使用未启用版本控制的存储桶或用于没有单独版本的对象。在这些情况下，Amazon S3 将返回版本 ID 为 `null` 的对象列表。

有关使用其他 AWS SDK 的信息，请参阅 AWS 开发人员中心。

Java

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListVersionsRequest;
import com.amazonaws.services.s3.model.S3VersionSummary;
import com.amazonaws.services.s3.model.VersionListing;

public class ListKeysVersioningEnabledBucket {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
```

```
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Retrieve the list of versions. If the bucket contains more versions
            // than the specified maximum number of results, Amazon S3 returns
            // one page of results per request.
            ListVersionsRequest request = new ListVersionsRequest()
                    .withBucketName(bucketName)
                    .withMaxResults(2);
            VersionListing versionListing = s3Client.listVersions(request);
            int numVersions = 0, numPages = 0;
            while (true) {
                numPages++;
                for (S3VersionSummary objectSummary :
versionListing.getVersionSummaries()) {
                    System.out.printf("Retrieved object %s, version %s\n",
                            objectSummary.getKey(),
                            objectSummary.getVersionId());
                    numVersions++;
                }
                // Check whether there are more pages of versions to retrieve. If
                // there are, retrieve them. Otherwise, exit the loop.
                if (versionListing.isTruncated()) {
                    versionListing =
s3Client.listNextBatchOfVersions(versionListing);
                } else {
                    break;
                }
            }
            System.out.println(numVersions + " object versions retrieved in " +
numPages + " pages");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        }
    }
}
```

.NET

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ListObjectsVersioningEnabledBucketTest
    {
        static string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main(string[] args)
        {
            s3Client = new AmazonS3Client(bucketRegion);
            GetObjectListWithAllVersionsAsync().Wait();
        }

        static async Task GetObjectListWithAllVersionsAsync()
        {
            try
            {
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    // You can optionally specify key name prefix in the request
                    // if you want list of object versions of a specific object.
```

```
                    // For this example we limit response to return list of 2
 versions.
                    MaxKeys = 2
                };
                do
                {
                    ListVersionsResponse response = await
 s3Client.ListVersionsAsync(request);
                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                } while (request != null);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

使用 REST API

Example — 列出存储桶中的所有对象版本

要列出存储桶中所有对象的所有版本，请在 versions 请求中使用 GET Bucket 子资源。Amazon S3 最多能检索 1000 个对象，且每个对象版本都将计为一个完整的对象。因此，如果存储桶包含两个键（例如 photo.gif 和 picture.jpg），并且第一个键有 990 个版本，第二个键有 400 个版本，则单个请求将检索 photo.gif 的所有 990 个版本，另加 picture.jpg 的最近 10 个版本。

Amazon S3 将按照存储的顺序返回对象版本，最先返回最近存储的版本。

在 GET Bucket 请求中，包含 versions 子资源。

```
GET /?versions HTTP/1.1
Host: bucketName.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

Example — 检索键的所有版本

要检索对象版本的子集，请为 GET Bucket 使用请求参数。有关更多信息，请参阅 GET Bucket。

1. 将 prefix 参数设置为您要检索的对象的键。
2. 使用 GET Bucket 子资源和 versions 发送 prefix 请求。

   ```
   GET /?versions&prefix=objectName HTTP/1.1
   ```

Example — 使用前缀检索对象

以下示例将检索其键是 myObject 或由它开头的对象。

```
GET /?versions&prefix=myObject HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

您可以使用其他请求参数来检索对象的所有版本的子集。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 GET Bucket。

Example — 在响应截断时检索其他对象的列表

如果可以在 GET 请求中返回的对象数量超过 max-keys 的值，则响应将包含 <isTruncated>true</isTruncated>，并包含满足该请求但不会返回的第一个键 (在 NextKeyMarker 中) 和第一个版本 ID (在 NextVersionIdMarker 中)。您可以在后续请求中将这些返回的值用作开始位置，以检索满足 GET 请求的其他对象。

使用以下过程从存储桶中检索满足原始 GET Bucket versions 请求的其他对象。有关 key-marker、version-id-marker、NextKeyMarker 和 NextVersionIdMarker 的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 GET Bucket。

以下是满足原始 GET 请求的其他响应：

- 将 key-marker 的值设置为在上一个响应中的 NextKeyMarker 中返回的键。
- 将 version-id-marker 的值设置为在上一个响应中的 NextVersionIdMarker 中返回的版本 ID。
- 使用 GET Bucket versions 和 key-marker 发送 version-id-marker 请求。

Example — 检索以指定的键和版本 ID 开始的对象

```
GET /?versions&key-marker=myObject&version-id-marker=298459348571 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

使用 AWS CLI

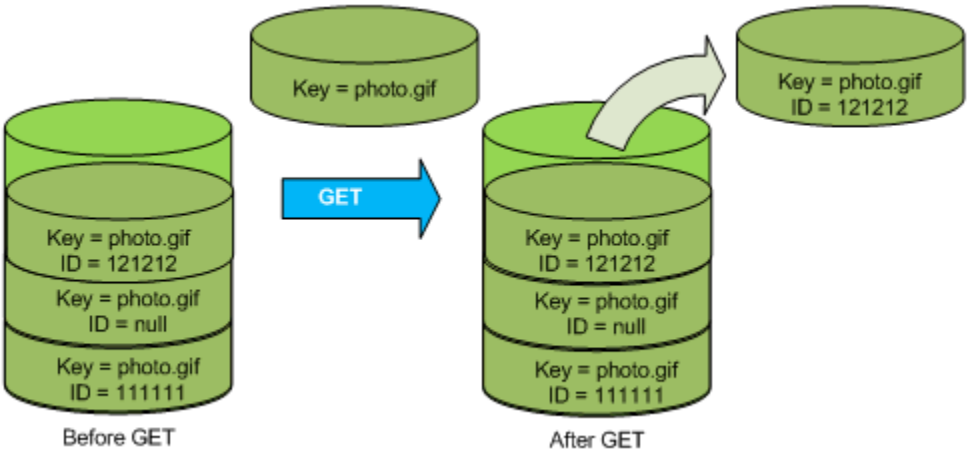
以下命令返回有关存储桶中的对象的所有版本的元数据。

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket1
```

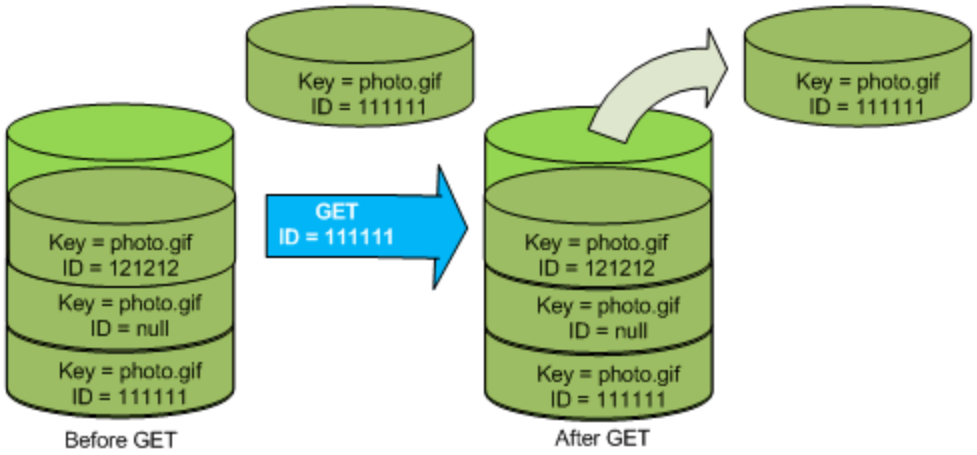
有关 list-object-versions 的更多信息，请参阅《AWS CLI 命令参考》中的 list-object-versions。

从启用了版本控制的存储桶中检索对象版本

Amazon S3 中的版本控制是在相同的存储桶中保留对象的多个变量的方法。简单 GET 请求将检索对象的当前版本。下图显示 GET 如何返回 photo.gif 对象的当前版本。



要检索特定版本，您需要指定其版本 ID。下图显示 `GET versionId` 请求检索对象的指定版本 (不一定是当前版本)。



您可以使用控制台、AWS SDK 或 REST API 在 Amazon S3 中检索对象版本。

> **Note**
>
> 要访问早于 300 个版本的对象版本，必须使用 AWS CLI 或对象的 URL。

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择包含对象的存储桶的名称。
3. 在 Objects（对象）列表中，请选择对象的名称。

4. 请选择 Versions（版本）。

   Amazon S3 显示了对象的所有版本。

5. 选中要检索的版本的 Version ID（版本 ID）旁的复选框。
6. 依次选择 Actions（操作）、Download（下载），然后保存对象。

您也可以在对象概述面板中查看、下载和删除对象版本。有关更多信息，请参阅 在 Amazon S3 控制台中查看对象概述。

> **Important**
>
> 仅当对象已作为最新 (当前) 版本删除时，您才能取消删除它。您无法取消删除已删除对象的早期版本。有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

使用 AWS SDK

在不受版本控制和启用了版本控制的存储桶中上传对象的示例是相同的。但是，对于启用了版本控制的存储桶，Amazon S3 会分配版本号。否则，版本号为空。

有关使用适用于 Java、.NET 和 PHP 的 AWS SDK 下载对象的示例，请参阅下载对象。

有关使用适用于 .NET 和 Rust 的 AWS SDK 列出对象版本的示例，请参阅列出 Amazon S3 存储桶中对象的版本。

使用 REST API

检索特定对象版本的步骤

1. 将 `versionId` 设置为您要检索的对象的版本 ID。
2. 发送 `GET Object versionId` 请求。

Example — 检索受版本控制的对象

以下请求将检索 `L4kqtJlcpXroDTDmpUMLUo` 的版本 `my-image.jpg`。

```
GET /my-image.jpg?versionId=L4kqtJlcpXroDTDmpUMLUo HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

您可以仅检索对象的元数据（而不是内容）。有关信息，请参阅 the section called “检索版本元数据”。

有关还原早期对象版本的信息，请参阅 the section called “还原早期版本”。

检索对象版本的元数据

如果您仅想检索对象的元数据 (而不是其内容)，您可以使用 HEAD 操作。默认情况下，您将获得最新版本的元数据。要检索特定对象版本的元数据，需要指定其版本 ID。

检索对象版本的元数据的步骤

1. 将 `versionId` 设置为您要检索其元数据的对象的版本 ID。
2. 发送 `HEAD Object versionId` 请求。

Example — 检索受版本控制的对象的元数据

以下请求将检索 `my-image.jpg` 的版本 `3HL4kqCxf3vjVBH40Nrjfkd` 的元数据。

```
HEAD /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

以下显示了示例响应。

```
HTTP/1.1 200 OK
x-amz-id-2: ef8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
x-amz-version-id: 3HL4kqtJlcpXroDTDmjVBH40Nrjfkd
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

还原早期版本

您可以使用版本控制来检索对象的早期版本。有两种方法可执行该操作：

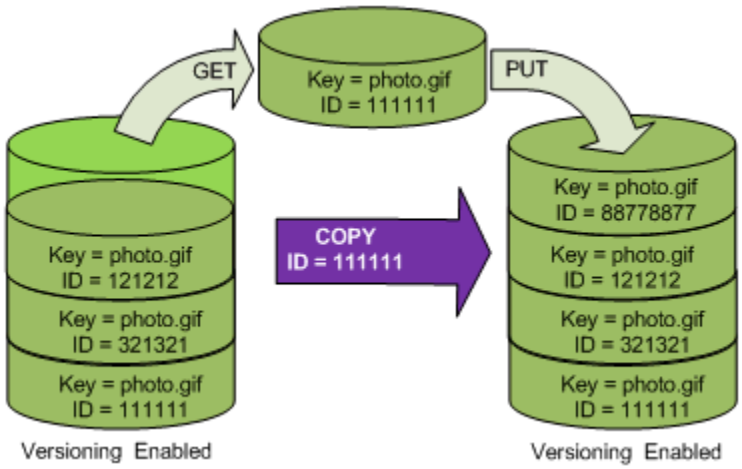
- 将对象的早期版本复制到同一存储桶中。

复制的对象将成为该对象的当前版本，且所有对象版本都保留。

- 永久删除对象的当前版本。

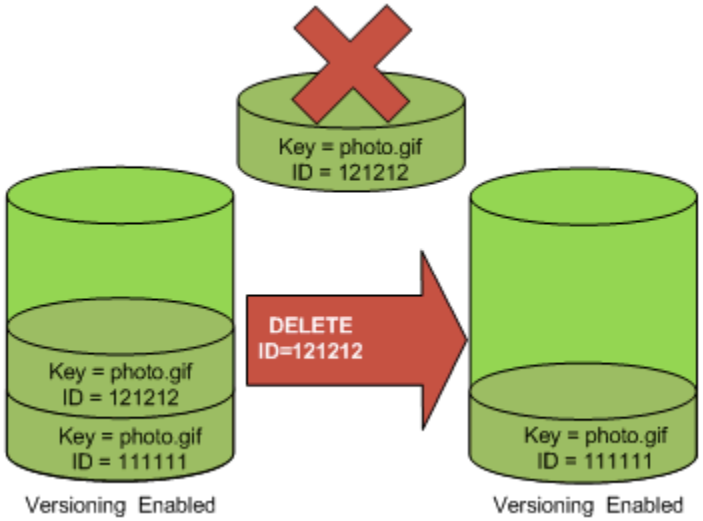
  当您删除当前对象版本时，实际上会将先前版本转换为该对象的当前版本。

由于所有对象版本都保留，所以您可以通过将对象的特定版本复制到同一存储桶，使任何较早版本成为当前版本。在下图中，源对象（ID = 111111）已复制到同一存储桶中。Amazon S3 将提供一个新 ID (88778877)，且它将成为该对象的当前版本。因此，该存储桶同时具有原始对象版本（111111）及其副本（88778877）。有关获取先前版本，然后将其上传以使其成为当前版本的更多信息，请参阅从启用了版本控制的存储桶中检索对象版本和上传对象。



随后，GET 将检索版本 88778877。

下图显示了如何删除对象的当前版本 (121212)，该操作保留早期版本 (111111) 作为当前版本。有关删除对象的更多信息，请参阅删除单个对象。



随后，GET 将检索版本 111111。

> **Note**
>
> 要批量还原对象版本，可以使用 CopyObject 操作。CopyObject 操作会复制清单中指定的每个对象。但请注意，不必以对象在清单中出现的相同顺序复制这些对象。对于受版本控制的存储桶，如果保留当前/非当前版本顺序很重要，您应首先复制所有非当前版本。然后，在第一个任务完成后，在后续任务中复制当前版本。

还原以前的对象版本

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择包含对象的存储桶的名称。
3. 在 Objects（对象）列表中，请选择对象的名称。
4. 请选择 Versions（版本）。

   Amazon S3 显示了对象的所有版本。
5. 选中要检索的版本的 Version ID（版本 ID）旁的复选框。
6. 依次选择 Actions（操作）、Download（下载），然后保存对象。

您也可以在对象概述面板中查看、下载和删除对象版本。有关更多信息，请参阅 在 Amazon S3 控制台中查看对象概述。

> **Important**
>
> 仅当对象已作为最新 (当前) 版本删除时，您才能取消删除它。您无法取消删除已删除对象的早期版本。有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

使用 AWS SDK

有关使用其他 AWS SDK 的信息，请参阅 AWS 开发人员中心。

Python

以下 Python 代码示例通过删除指定回滚版本之后的所有版本，来还原版本控制对象的以前版本。

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
```

```
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

从启用了版本控制的存储桶中删除对象版本

您在需要时随时可以从 Amazon S3 存储桶中删除对象版本。您还可为具有明确定义的生命周期的对象定义生命周期配置规则，请求 Amazon S3 使当前对象版本过期，或者永久删除非当前对象版本。当存储桶已启用版本控制或者版本控制已暂停时，生命周期配置操作的工作方式如下：

- `Expiration` 操作适用于当前对象版本。Amazon S3 通过添加删除标记将当前版本作为非当前版本保留（而不是删除当前对象版本），然后删除标记将成为当前版本。
- `NoncurrentVersionExpiration` 操作适用于非当前对象版本，Amazon S3 会永久删除这些对象版本。无法恢复永久删除的对象。

有关 S3 生命周期的更多信息，请参阅管理存储生命周期和S3 生命周期配置的示例。

要查看您的存储桶有多少个当前和非当前对象版本，您可以使用 Amazon S3 Storage Lens 存储统计管理工具指标。S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。有关更多信息，请参阅使用 S3 Storage Lens 存储统计管理工具优化存储成本。有关指标的完整列表，请参阅 S3 Storage Lens 存储统计管理工具指标词汇表。

> **Note**
>
> 正常的 Amazon S3 费率适用于存储和传输的每个对象版本，包括非当前对象版本。有关更多信息，请参阅 Amazon S3 定价。

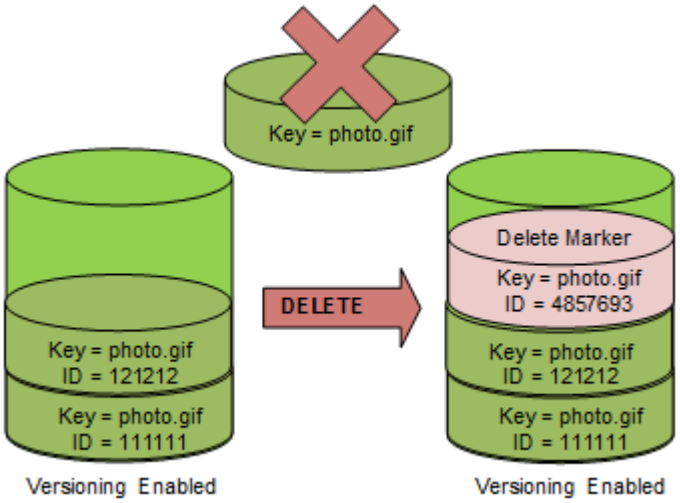
删除请求使用案例

`DELETE` 请求具有以下使用案例：

- 启用版本控制后，简单 `DELETE` 无法永久删除对象。（简单 `DELETE` 请求是指未指定版本 ID 的请求。）Amazon S3 将在存储桶中插入删除标记，该删除标记将成为对象的当前版本并具有新的 ID。

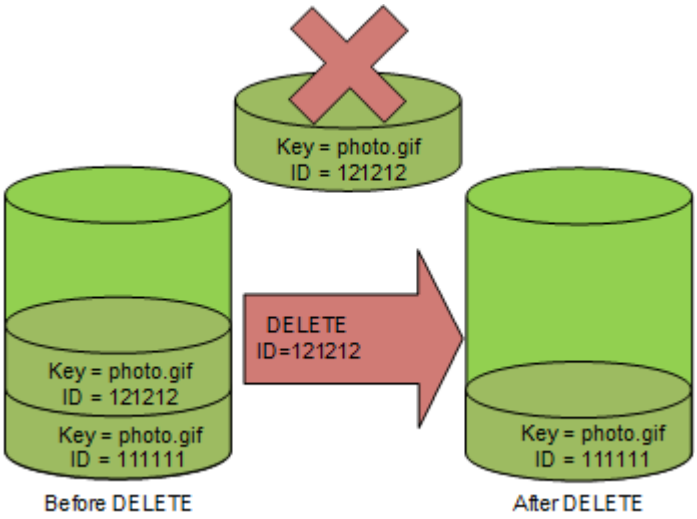
当您尝试对当前版本为删除标记的对象执行 GET 操作时，Amazon S3 将该对象作为已删除对象对待（即使它尚未被擦除），并返回 404 错误。有关更多信息，请参阅 使用删除标记。

下图显示简单 DELETE 实际上不会删除指定的对象。但是，Amazon S3 将插入一个删除标记。



- 要永久删除受版本控制的对象，您必须使用 DELETE Object versionId。

下图显示删除指定的对象版本将永久删除该对象。



要删除对象版本

您可以使用控制台、AWS SDK 或 REST API 或 AWS Command Line Interface 删除 Amazon S3 中的对象版本。

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择包含对象的存储桶的名称。
3. 在 Objects（对象）列表中，请选择对象的名称。
4. 请选择 Versions（版本）。

   Amazon S3 显示了对象的所有版本。
5. 选中要检索的版本的版本 ID 旁的复选框。
6. 请选择 Delete（删除）。
7. 在 Permanently delete objects?（永久删除对象？）中，输入 **`permanently delete`**。

   > **Warning**
   >
   > 永久删除对象版本时，该操作将无法撤消。

8. 请选择 Delete objects（删除对象）。

   Amazon S3 将删除对象版本。

使用 AWS SDK

有关使用适用于 Java、.NET 和 PHP 的 AWS SDK 删除对象的示例，请参阅 删除 Amazon S3 对象。在不受版本控制和启用了版本控制的存储桶中删除对象的示例是相同的。但是，对于启用了版本控制的存储桶，Amazon S3 会分配版本号。否则，版本号为空。

有关使用其他 AWS SDK 的信息，请参阅 AWS 开发人员中心。

Python

以下 Python 代码示例通过删除版本控制对象的所有版本永久删除了该对象。

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.
```

```
    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

使用 REST API

删除对象的特定版本

- 在 DELETE 中，指定版本 ID。

Example — 删除特定版本

以下示例删除了 photo.gif 的版本 UIORUnfnd89493jJFJ。

```
DELETE /photo.gif?versionId=UIORUnfnd89493jJFJ HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: text/plain
Content-Length: 0
```

使用 AWS CLI

以下命令从名为 *amzn-s3-demo-bucket1* 的存储桶中删除名为 test.txt 的对象。要删除特定版本的对象，您必须是存储桶拥有者，并且您必须使用版本 Id 子资源。

```
aws s3api delete-object --bucket amzn-s3-demo-bucket1 --key test.txt --version-
id versionID
```

有关 delete-object 的更多信息，请参阅《AWS CLI 命令参考》中的 delete-object。

有关删除对象版本的更多信息，请参阅以下主题：

- 使用删除标记
- 删除删除标记以使旧版本变为当前版本
- 从启用了 MFA 删除的存储桶中删除对象

使用删除标记

Amazon S3 中的删除标记是用于在简单 DELETE 请求中指定的、启用了版本控制的对象的占位符（或标记）。简单 DELETE 请求是指未指定版本 ID 的请求。因为对象位于已启用版本控制的存储桶中，所以不能删除该对象。但是，删除标记可以使 Amazon S3 的行为类似于对象已被删除。在删除标记上可以使用 Amazon S3 API DELETE 调用。为此，您必须使用具有适当权限的 AWS Identity and Access Management (IAM) 用户或角色来发出 DELETE 请求。

与任何其他对象一样，删除标记同样有键名（或键）和版本 ID。但是，删除标记在以下方面与其他对象不同：

- 删除标记没有与之关联的数据。
- 删除标记不与访问控制列表 (ACL) 值关联。
- 如果您针对删除标记发出 GET 请求，则 GET 请求不会检索任何内容，因为删除标记没有数据。具体而言，如果您的 GET 请求未指定 `versionId`，则会收到 404（未找到）错误。

删除标记会对 Amazon S3 中的存储产生极少的费用。删除标记的存储大小等于删除标记键名的大小。键名是 Unicode 字符序列。对于名称中的每个字符，键名的 UTF-8 编码将 1 到 4 字节的存储添加到存储桶中。删除标记存储在 S3 Standard 存储类中。

如果您想知道自己有多少个删除标记以及它们存储在哪个存储类中，可以使用 Amazon S3 Storage Lens 存储统计管理工具。有关更多信息，请参阅使用 Amazon S3 Storage Lens 存储统计管理工具评估您的存储活动和使用情况 和Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

有关键名称的更多信息，请参阅 创建对象键名称。有关将删除标记删除的信息，请参阅 管理删除标记。

仅 Amazon S3 可以创建删除标记，且当您在已启用版本控制或已暂停版本控制的存储桶的对象上发送 `DeleteObject` 请求时执行此操作。在 DELETE 请求中指定的对象不会实际删除。而是使删除标记成为对象的当前版本。该对象的键名（键）将成为删除标记的键。

如果您获取对象而未在请求中指定 `versionId`，如果其当前版本是一个删除标记，则 Amazon S3 的响应如下：

- 404（未找到）错误
- 响应标头，x-amz-delete-marker: true

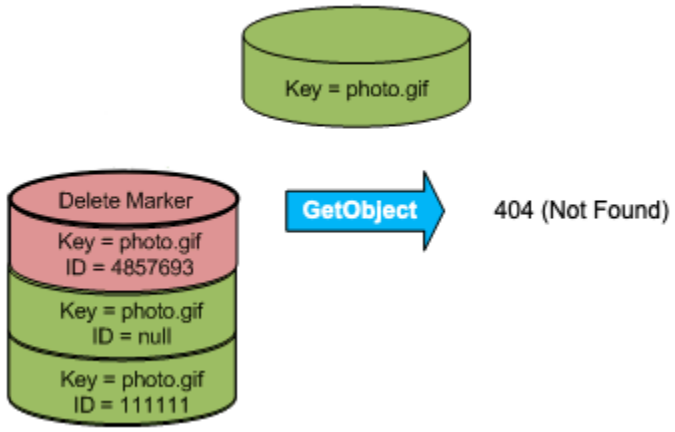
当您通过在请求中指定 versionId 来获取对象时，如果其当前版本是一个删除标记，则 Amazon S3 的响应如下：

- 405（不允许此方法）错误
- 响应标头，x-amz-delete-marker: true
- 响应标头，Last-Modified: timestamp（仅在使用 HeadObject 或 GetObject API 操作时）

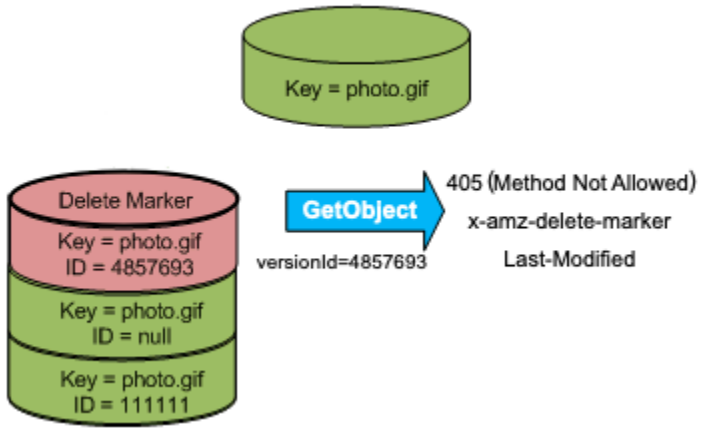
x-amz-delete-marker: true 响应标头告知您所访问的对象是删除标记。此响应标头永远不会返回 false，因为当值为时 false，对象的当前版本或指定版本不是删除标记。

Last-Modified 响应标头提供删除标记的创建时间。

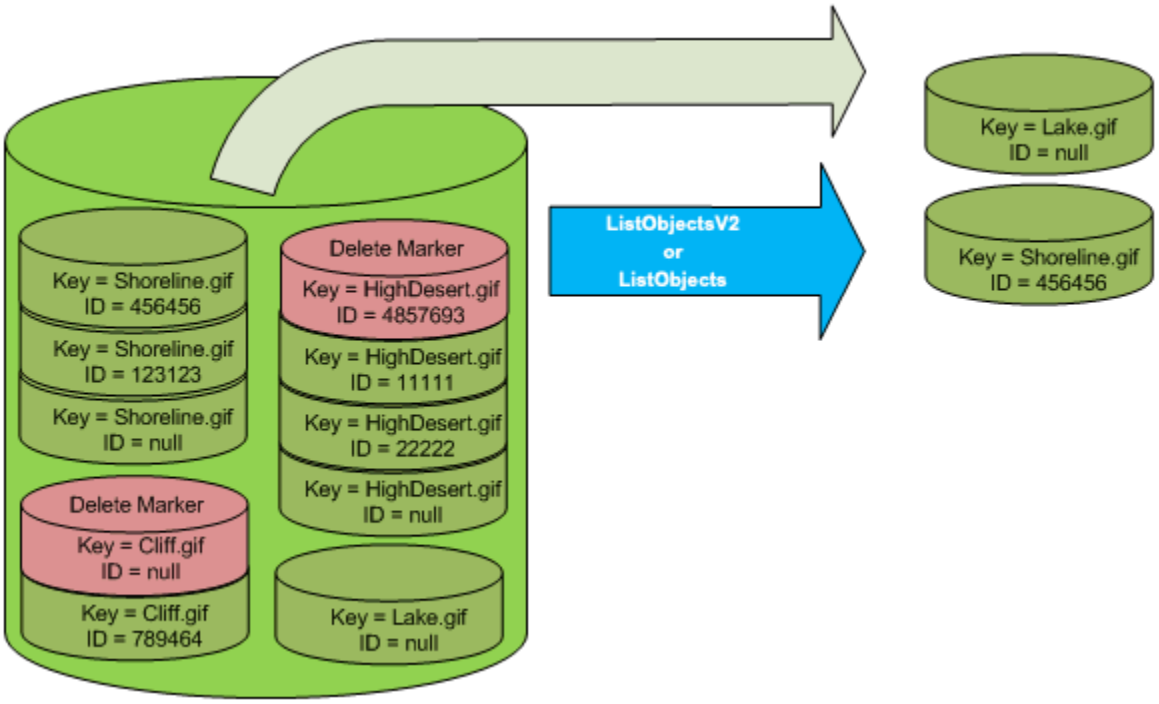
下图显示了对当前版本为删除标记的对象进行 GetObject API 调用时，该调用如何使用 404（未找到）错误进行响应且响应标头中包括 x-amz-delete-marker: true。



如果您通过在请求中指定 versionId 来对对象进行 GetObject 调用，并且指定的版本是删除标记，则 Amazon S3 会以 405（不允许方法）错误进行响应，且响应标头包括 x-amz-delete-marker: true 和 Last-Modified: timestamp。



列出删除标记（和对象的其他版本）的唯一方法在 `versions` 请求中使用 ListObjectVersions 子资源。下图显示 ListObjectsV2 或 ListObjects 请求不会返回其当前版本为删除标记的对象。



管理删除标记

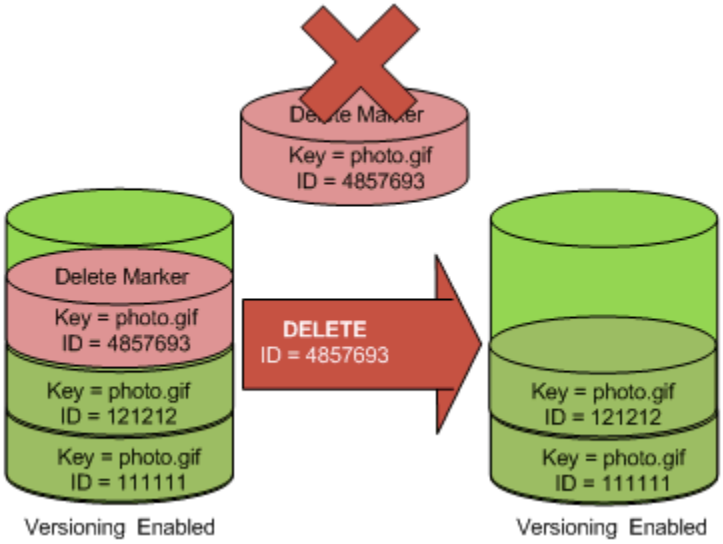
配置生命周期以自动清理过期的删除标记

过期的对象删除标记是指，删除所有对象版本而只保留一个删除标记。如果生命周期配置设置为删除当前版本，或显式设置了 `ExpiredObjectDeleteMarker` 操作，则 Amazon S3 会删除过期对象的删除标记。有关示例，请参阅示例 7：移除过期对象删除标记。

删除删除标记以使旧版本变为当前版本

当您删除启用了版本控制的存储桶中的某个对象时，所有版本都将保留在存储桶中并且 Amazon S3 将为该对象创建一个删除标记。要取消删除该对象，您必须删除此删除标记。有关版本控制和删除标记的更多信息，请参阅在 S3 存储桶中使用版本控制。

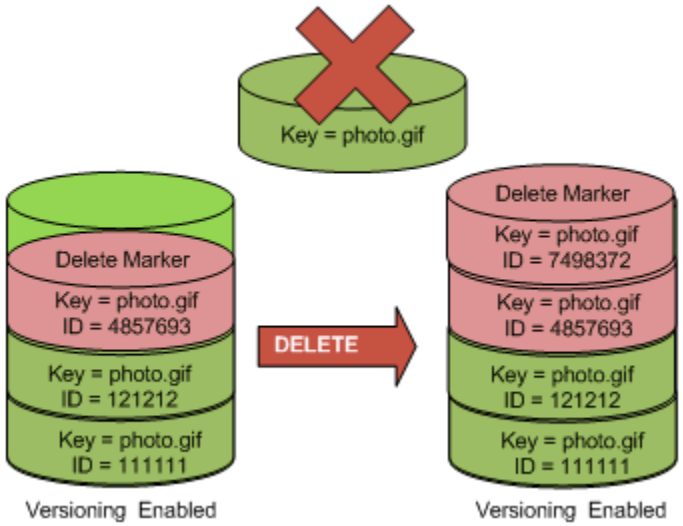
要永久删除“删除标记”，必须在 DeleteObject versionId 请求中包含其版本 ID。下图显示了 DeleteObject versionId 请求如何永久删除“删除标记”。



去除删除标记的效果是：简单的 GET 请求现在将检索对象的当前版本 ID（121212）。

> **Note**
>
> 如果在当前版本是一个删除标记（未指定删除标记的版本 ID）的情况下使用 DeleteObject 请求，Amazon S3 将不会删除该删除标记，而是对另一个删除标记执行 PUTs 请求。

要删除具有 NULL 版本 ID 的删除标记，必须在 DeleteObject 请求中将 NULL 作为版本 ID 传递。下图显示了在没有版本 ID（当前版本是一个删除标记）的情况下如何发出简单的 DeleteObject 请求，未删除任何内容，而是再添加一个具有唯一版本 ID (7498372) 的删除标记。



使用 S3 控制台

按照以下步骤，从 S3 存储桶中恢复文件夹之外的已删除对象，包括这些文件夹中的对象。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择您想要的存储桶的名称。
3. 要查看存储桶中对象的版本列表，请选择列出版本开关。您将能够看到已删除对象的删除标记。
4. 要取消删除对象，您必须删除掉删除标记。选中要恢复的对象的删除标记旁边的复选框，然后选择 Delete (删除)。
5. 在 Delete objects (删除对象) 页面上确认删除。
   a. 对于 Permanently delete objects? (永久删除对象？) ，输入 **`permanently delete`**。
   b. 请选择 Delete objects (删除对象) 。

> **Note**
>
> 您无法使用 Amazon S3 控制台取消删除文件夹。您必须使用 AWS CLI 或 SDK。例如，请参阅 AWS 知识中心中的如何检索启用版本控制的存储桶中已被删除的 Amazon S3 对象？。

使用 REST API

永久删除“删除标记”的步骤

1. 将 `versionId` 设置为要删除的删除标记的版本 ID。
2. 发送 `DELETE Object versionId` 请求。

Example — 删除“删除标记”

以下示例删除用于 `photo.gif` 版本 4857693 的删除标记。

```
DELETE /photo.gif?versionId=4857693 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

删除“删除标记”时，Amazon S3 将在响应中包含以下内容：

```
204 NoContent
x-amz-version-id: versionID
x-amz-delete-marker: true
```

使用 AWS SDK

有关使用其他 AWS SDK 的信息，请参阅 AWS 开发人员中心。

Python

以下 Python 代码示例演示了如何从对象中删除“删除标记”，从而使最新的非当前版本成为对象的当前版本。

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.
```

```
    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

从启用了 MFA 删除的存储桶中删除对象

如果存储桶的版本控制配置已启用 MFA 删除，则存储桶拥有者必须在请求中包含 x-amz-mfa 请求标头，才能永久删除对象版本或更改该存储存储桶的版本控制状态。包含 x-amz-mfa 的请求必须使用 HTTPS。

标头的值由身份验证设备的序列号、空格，以及在该设备上显示的身份验证代码组成。如果您没有包含此请求标头，则请求失败。

有关身份验证设备的更多信息，请参阅多重身份验证。

Example — 从启用了 MFA 删除的存储桶中删除对象

以下示例删除了在存储桶中配置为启用 MFA 删除的 my-image.jpg（具有指定版本）。

请注意 *[SerialNumber]* 和 *[AuthenticationCode]* 之间的空格。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 DeleteObject。

```
DELETE /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTPS/1.1
Host: bucketName.s3.amazonaws.com
x-amz-mfa: 20899872 301749
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

有关启用 MFA 删除的更多信息，请参阅 配置 MFA 删除。

配置受版本控制的对象权限

Amazon S3 中的对象的权限在版本级别设置。每个版本都有自己的对象拥有者。创建对象版本的 AWS 账户 是拥有者。因此，您可以为同一对象的不同版本设置不同的权限。要这样做，您必须指定要在 PUT Object versionId acl 请求中设置其权限的对象的版本 ID。有关详细描述和关于使用 ACL 的说明，请参阅 Amazon S3 的身份和访问管理。

Example — 为对象版本设置权限

以下请求将被授权者（BucketOwner@amazon.com）对键（my-image.jpg）和版本 ID（3HL4kqtJvjVBH40Nrjfkd）的权限设置为 FULL_CONTROL。

```
PUT /my-image.jpg?acl&versionId=3HL4kqtJvjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
Content-Length: 124

<AccessControlPolicy>
  <Owner>
    <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>mtd@amazon.com</DisplayName>
  </Owner>
```

```
 <AccessControlList>
   <Grant>
     <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
       <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
       <DisplayName>BucketOwner@amazon.com</DisplayName>
     </Grantee>
     <Permission>FULL_CONTROL</Permission>
   </Grant>
 </AccessControlList>
 </AccessControlPolicy>
```

同样地，要获取特定对象版本的权限，必须在 `GET Object versionId acl` 请求中指定它的版本 ID。您需要包含版本 ID，因为在默认情况下，`GET Object acl` 将返回对象的当前版本的权限。

Example — 检索用于指定对象版本的权限

在以下示例中，Amazon S3 将返回对于键（`my-image.jpg`）和版本 ID（`DVBH40Nr8X8gUMLUo`）的权限。

```
GET /my-image.jpg?versionId=DVBH40Nr8X8gUMLUo&acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU
```

有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 GetObjectAcl。

使用已暂停版本控制的存储桶中的对象

在 Amazon S3 中，您可以暂停版本控制，以停止在存储桶中累积同一对象的新版本。您可能这样做是因为您只想在存储桶中使用单个版本的对象。或者，您可能不想为多个版本累积费用。

暂停版本控制时，存储桶中的现有对象不会更改。更改的是 Amazon S3 在以后的请求中处理对象的方式。本部分中的主题介绍了暂停版本控制的存储桶中的各种对象操作，包括添加、检索和删除对象。

有关 S3 版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。有关检索对象版本的更多信息，请参阅 从启用了版本控制的存储桶中检索对象版本。

主题

- 将对象添加到已暂停版本控制的存储桶
- 从已暂停版本控制的存储桶检索对象

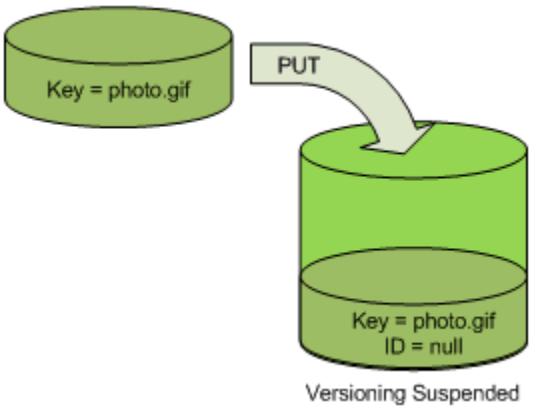
- 从已暂停版本控制的存储桶中删除对象

将对象添加到已暂停版本控制的存储桶

您可以将对象添加到 Amazon S3 中已暂停版本控制的存储桶中，以便创建具有空版本 ID 的对象，或者覆盖具有匹配版本 ID 的任何对象版本。

您在存储桶上暂停了版本控制后，Amazon S3 会自动将 `null` 版本 ID 添加到之后存储在该存储段中的每个后续对象（使用 PUT、POST 或 `CopyObject`）。

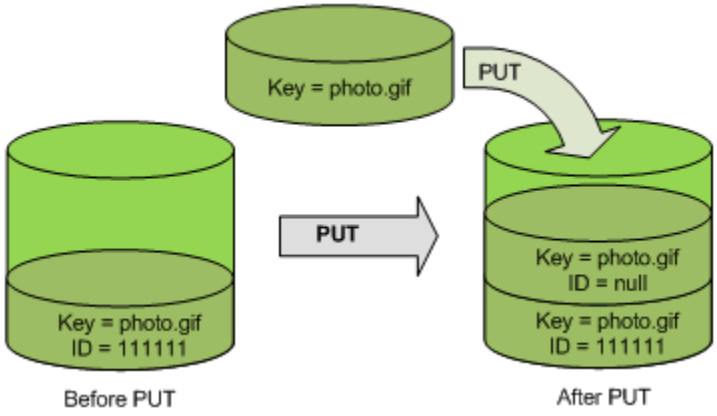
下图显示了当将对象添加到暂停版本控制的存储桶时，Amazon S3 将如何向该对象添加 `null` 的版本 ID。



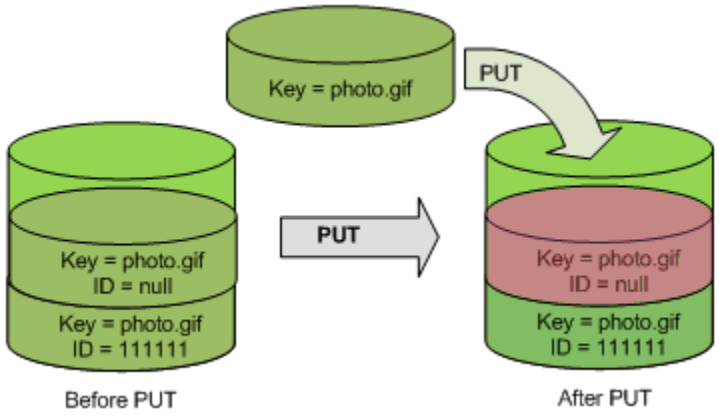
如果存储桶中已存在空版本，且您使用同一键添加了其他对象，则添加的对象将覆盖原始的空版本。

如果存储桶中存在受版本控制的对象，则使用 PUT 存储的版本将成为该对象的当前版本。下图显示了如何将对象添加到包含受版本控制的对象（不会覆盖已存在于该存储桶中的对象）的存储桶。

在这种情况下，版本 111111 已存在于该存储桶中。Amazon S3 会将空的版本 ID 附加到所添加的对象，并将其存储在存储桶中。版本 111111 不会被覆盖。



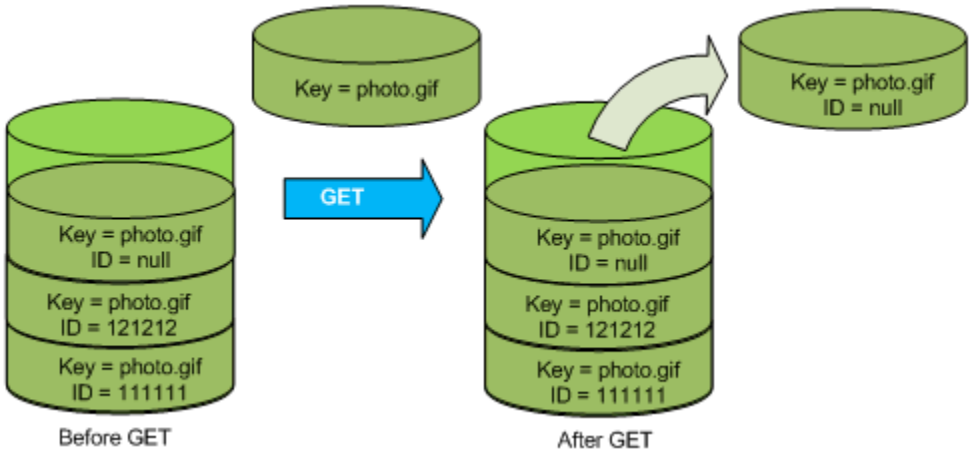
如果存储桶中已存在空版本，则该空版本将被覆盖，如下图所示。



虽然空版本的键和版本 ID（null）在 PUT 之前和之后都相同，但是原来存储在存储桶中的空版本的内容将替换为该存储桶中对象 PUT 的内容。

从已暂停版本控制的存储桶检索对象

无论是否在存储桶上启用版本控制，GET Object 请求都将返回对象的当前版本。下图显示了简单 GET 将如何返回对象的当前版本。



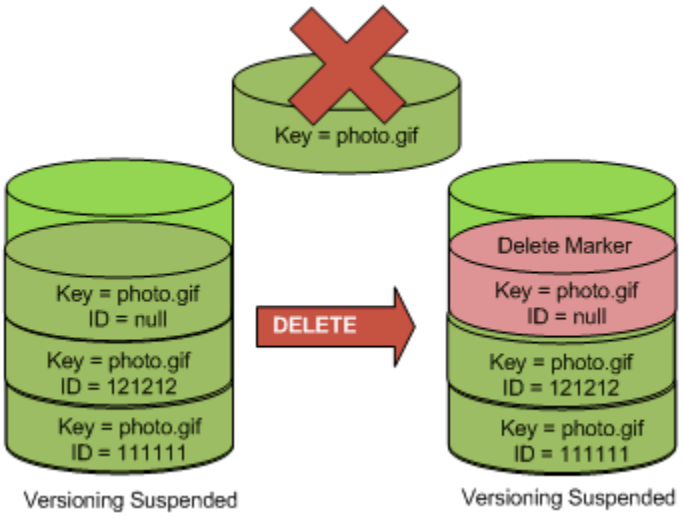
从已暂停版本控制的存储桶中删除对象

您可以将对象从暂停版本控制的存储桶中删除，以便删除具有空版本 ID 的对象。

如果某个存储桶的版本控制已暂停，DELETE 请求：

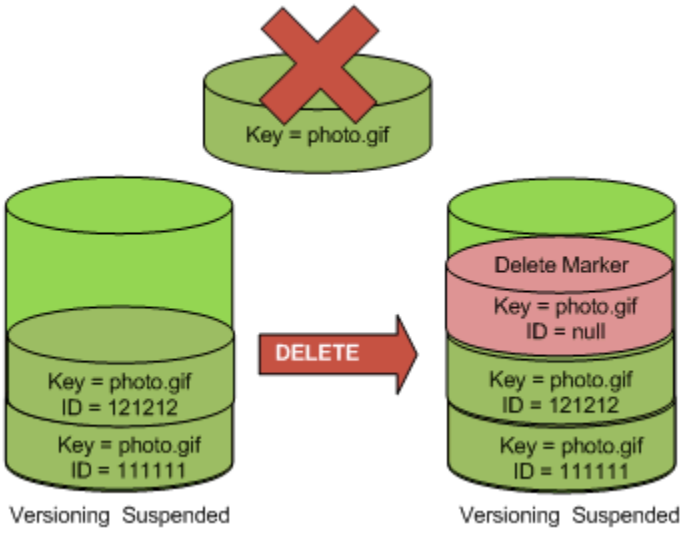
- 可以仅删除其版本 ID 为 null 的对象。
- 如果存储桶中没有对象的空版本，则不删除任何内容。
- 将删除标记插入到存储桶。

下图显示了简单的 DELETE 如何删除空版本。（简单 DELETE 请求是指未指定版本 ID 的请求。）Amazon S3 在其版本 ID 为 null 的位置插入一个删除标记。

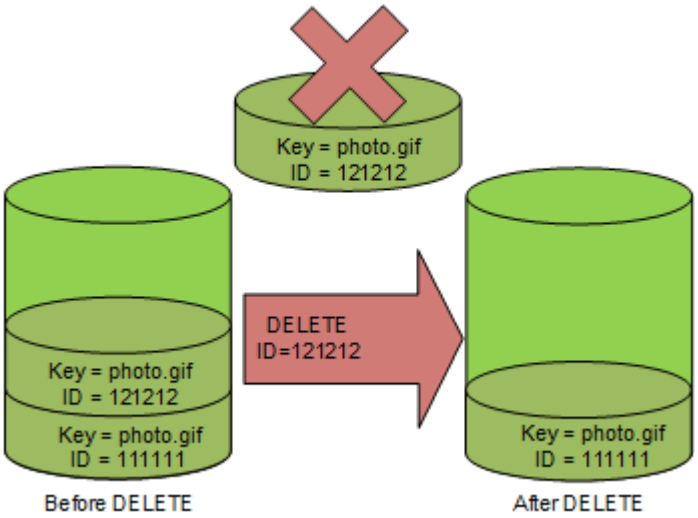


请记住，删除标记不包含任何内容，所以您会在删除标记替换空版本时丢失该空版本的内容。

下图显示不具有空版本的存储桶。在这种情况下，DELETE 不会删除任何内容；Amazon S3 仅插入删除标记。



即使是在已暂停版本控制的存储桶中，存储桶拥有者也可以通过在 DELETE 请求中包括版本 ID 来永久删除指定版本。下图显示删除指定的对象版本将永久删除该版本的对象。只有存储桶拥有者可以删除指定的对象版本。



使用适用于 Amazon S3 的 AWS Backup

Amazon S3 与 AWS Backup 本机集成，是一项完全托管式、基于策略的服务，您可以使用它来集中定义备份策略，以保护 Amazon S3 中的数据。在定义备份策略并将 Amazon S3 资源分配给策略之后，AWS Backup 自动创建 Amazon S3 备份，并将备份安全存储在备份计划中指定的加密备份文件库中。

当将 AWS Backup 用于 Amazon S3 时，您可以执行以下操作：

- 创建连续备份和定期备份。连续备份对于时间点还原很有用，定期备份对于满足您的长期数据保留需求很有用。
- 通过集中配置备份策略，自动化备份计划和保留。
- 将 Amazon S3 数据备份还原到您指定的时间点。

您可以连同 AWS Backup 使用 S3 版本控制和 S3 复制来帮助从意外删除中恢复并执行自己的自我恢复操作。

先决条件

您必须先在桶上激活 S3 版本控制，AWS Backup 才可以对它进行备份。

> **Note**
>
> 我们建议您为正在备份的启用版本控制的桶设置生命周期过期规则。如果您没有设置生命周期过期的时段，则 Amazon S3 存储成本可能会增加，原因是 AWS Backup 将保留 Amazon S3 数据的所有版本。

开始使用

要开始对 Amazon S3 使用 AWS Backup，请参阅《AWS Backup 开发人员指南》中的创建 Amazon S3 备份。

限制和局限性

要了解相关限制，请参阅《AWS Backup 开发人员指南》中的创建 Amazon S3 备份。

使用归档的对象

要降低不经常访问的对象的存储成本，您可以对此类对象进行归档。当您归档对象时，它会被移动到低成本存储中，这意味着您无法实时访问它。

尽管无法实时访问归档对象，但您可以在几分钟或几小时内还原它们，具体取决于存储类。您可以使用 Amazon S3 控制台、S3 批量操作、REST API、AWS SDK 和 AWS Command Line Interface（AWS CLI）还原归档的对象。有关说明，请参阅 恢复已归档的对象。

以下存储类或存储层中的 Amazon S3 对象已归档，无法实时访问：

- S3 Glacier Flexible Retrieval 存储类
- S3 Glacier Deep Archive 存储类
- S3 Intelligent-Tiering 归档访问层
- S3 Intelligent-Tiering 深度归档访问层

要还原归档对象，必须执行以下操作：

- 对于 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类中的对象，您必须启动还原请求，并等待对象的临时副本可用。当创建了还原对象的临时副本时，该对象的存储类保持不变。（HeadObject 或 GetObject API 操作请求会返回 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 作为存储类。）

- 对于 S3 Intelligent-Tiering 归档访问层和 S3 Intelligent-Tiering 深度归档访问层中的对象，您必须启动还原请求，并等待对象移动到频繁访问层。

有关所有 Amazon S3 存储类的比较情况的更多信息，请参阅使用 Amazon S3 存储类。有关 S3 Intelligent-Tiering 的更多信息，请参阅the section called “S3 Intelligent-Tiering 工作原理”。

从 S3 Glacier 还原对象

当您使用 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 时，Amazon S3 仅在指定的持续时间内恢复对象的临时副本。在此之后，它会删除还原的对象副本。您可以通过重新发出还原请求来修改已还原副本的到期时间。在这种情况下，Amazon S3 会更新相对于当前时间的有效期。

> **Note**
>
> 当您从 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 还原归档对象时，您需要同时为归档对象和临时还原的副本付费。有关定价的信息，请参阅 Amazon S3 定价。

从 S3 Intelligent-Tiering 还原对象

当您从 S3 Intelligent-Tiering 归档访问层或 S3 Intelligent-Tiering 深度归档访问层还原对象时，对象会移回到 S3 Intelligent-Tiering 频繁访问层中。如果对象在连续 30 天后仍未被访问，它会自动进入不频繁访问层。至少连续 90 天无访问后，对象将移动到 S3 Intelligent-Tiering 归档访问层。如果对象在至少连续 180 天后未被访问，则该对象将移动到深层归档访问层。

> **Note**
>
> 与 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类不同，S3 Intelligent-Tiering 对象的还原请求不接受该 `Days` 值。

结合使用 S3 批量操作与还原请求

要使用单个请求恢复多个 Amazon S3 对象，您可以使用 S3 分批操作。您为 S3 批量操作提供要操作的对象列表。S3 批量操作调用相应的 API 操作来执行指定的操作。单个批量操作任务可对包含 EB 级数据的数十亿个对象执行指定操作。

还原时间

Amazon S3 通过将还原请求中指定的天数与所请求的还原的完成时间相加来计算还原的对象副本的到期时间。然后，Amazon S3 将得出的时间舍入至第二天的午夜协调世界时（UTC)。例如，假设还原的对象副本是在 2012 年 10 月 15 日上午 10:30 UTC 创建的，并且还原时段被指定为 3 天。在这种情况下，还原的副本将于 2012 年 10 月 19 日 00:00 UTC 到期，届时 Amazon S3 将删除对象副本。

完成还原任务所需的时间取决于您使用的归档存储类或存储层和您指定的检索选项：加速（仅适用于 S3 Glacier Flexible Retrieval 和 S3 Intelligent-Tiering 归档访问）、标准或批量。有关更多信息，请参阅 归档检索选项。

使用 Amazon S3 事件通知，您可以在还原完成时得到通知。有关更多信息，请参阅 Amazon S3 事件通知。

主题

归档检索选项

以下是在 Amazon S3 中还原已归档对象时可用的检索选项：

- 加速 – 快速访问存储在 S3 Glacier Flexible Retrieval 存储类或 S3 Intelligent-Tiering 归档访问层中的数据。可以在偶尔需要紧急请求归档子集时使用此选项。对于除最大归档对象（250MB+）之外的所有其他归档对象，使用加速检索访问的数据通常在 1 到 5 分钟内可用。

  > **Note**
  >
  > 加速检索是一项高级特征，按加急请求和检索费率收费。
  > 有关 Amazon S3 定价的信息，请参阅 Amazon S3 定价。

  预配置容量帮助确保在您需要时，可以使用针对 S3 Glacier Flexible Retrieval 中的加速检索的检索容量。有关更多信息，请参阅 预调配容量。
- 标准 - 在数小时内访问您的任意归档对象。“标准”是未指定检索选项的检索请求的原定设置选项。对于存储在 S3 Glacier Flexible Retrieval 存储类或 S3 Intelligent-Tiering 归档访问层中的对象，标准检索通常在 3-5 小时内完成。对于存储在 S3 Glacier Deep Archive 存储类或 S3 Intelligent-Tiering 深

度归档访问层中的对象，这些检索通常会在 12 小时内完成。对于存储在 S3 Intelligent-Tiering 中的对象，标准检索是免费的。

> **Note**
>
> - 对于存储在 S3 Glacier Flexible Retrieval 存储类或 S3 Intelligent-Tiering 存档访问层中的对象，使用 S3 批量操作还原操作启动的标准检索通常在几分钟内开始，并在 3-5 小时内完成。
> - 对于 S3 Glacier Deep Archive 存储类或 S3 Intelligent-Tiering 深度存档访问层中的对象，使用批量操作还原操作启动的标准检索通常在 9 小时内开始，并在 12 小时内完成。

- 批量 – 使用 Amazon S3 Glacier 中成本最低的检索选项访问您的数据。通过批量检索，您能够以低廉的成本检索大量数据，甚至达到 PB 级。

  对于存储在 S3 Glacier Flexible Retrieval 存储类或 S3 Intelligent-Tiering 存档访问层中的对象，批量检索通常在 5-12 小时内完成。对于存储在 S3 Glacier Deep Archive 存储类或 S3 Intelligent-Tiering 深度存档访问层中的对象，这些检索通常会在 48 小时内完成。

  对于存储在 S3 Glacier Flexible Retrieval 或 S3 Intelligent-Tiering 存储类中的对象，批量检索是免费的。

下表总结了归档检索选项。有关定价的信息，请参阅 Amazon S3 定价。

要进行 `Expedited`、`Standard` 或 `Bulk` 检索，请将 RestoreObject REST API 操作请求中的 `Tier` 请求元素设置为您需要的选项，或 AWS Command Line Interface（AWS CLI）或 AWS SDK 中的等效选项。如果您购买了预配置容量，则所有加速检索都会通过您的预配置容量自动获得处理。

预调配容量

预调配容量的作用在于：当您从 S3 Glacier Flexible Retrieval 中进行加速检索时，它可帮助确保您在需要时获得检索容量。每个容量单位确保每 5 分钟至少可以执行 3 个加速检索，并提供高达 150MB/秒（MBps）的检索吞吐量。

如果您的工作负载要求在几分钟内对数据子集进行高度可靠和可预测的访问，应考虑购买预调配的检索容量。如果没有预配置容量，在高需求期间可能不会接受加速检索。如果您需要随时可以访问加速检索，建议您购买预配置检索容量。

预调配容量单位分配给 AWS 账户。因此，加速数据检索的请求者（而不是桶拥有者）应购买预调配容量单位。

您可以使用 Amazon S3 控制台、Amazon S3 Glacier 控制台、购买预调配容量 REST API 操作、AWS SDK 或 AWS CLI 购买预调配容量。有关预调配容量的定价信息，请参阅 Amazon S3 定价。

S3 Glacier 还原启动请求速率

当您为存储在 S3 Glacier Flexible Retrieval 或 S3 Glacir Deep Archive 存储类中的对象启动还原请求时，将为您的 AWS 账户应用检索请求限额。S3 Glacier 支持以每秒 1000 个事务的速率发出还原请求。如果超过此速率，则有效请求将受限制或被拒绝，而 Amazon S3 会返回 `ThrottlingException` 错误。

或者，您还可以使用 S3 批量操作通过单个请求检索存储在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 中的大量对象。有关更多信息，请参阅 对 Amazon S3 对象执行大规模批量操作。

恢复已归档的对象

以下存储类或存储层中的 Amazon S3 对象已归档，无法实时访问：

- S3 Glacier Flexible Retrieval 存储类
- S3 Glacier Deep Archive 存储类
- S3 Intelligent-Tiering 归档访问层
- S3 Intelligent-Tiering 深度归档访问层

无法立即访问存储在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中的 Amazon S3 对象。要访问这些存储类中的对象，您必须在指定的持续时间（天数）内将该对象的临时副本还原到其 S3 桶。如果您想要该对象的永久副本，请还原该对象，然后在 Amazon S3 桶中创建该对象的一个副本。Amazon S3 控制台不支持复制已还原的对象。对于这种类型的复制操作，请使用 AWS Command Line Interface（AWS CLI）、AWS SDK 或 REST API。除非您制作副本并更改其存储类，否则对象仍将存储在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中。有关使用这些存储类的信息，请参阅极少访问的对象的存储类。

要访问 S3 Intelligent-Tiering 归档访问层和深度归档访问层中的对象，您必须启动还原请求，并等待对象移动到频繁访问层。从归档访问层或深度归档访问层进行还原时，对象会转换回 频繁访问层中。有关使用这些存储类的信息，请参阅用于自动优化访问模式不断变化或未知的数据的存储类。

有关归档对象的一般信息，请参阅使用归档的对象。

Note

- 当您从 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类还原存档对象时，您需要同时为存档对象和临时还原的副本付费。
- 从 S3 Intelligent-Tiering 还原对象时，标准检索或批量检索不收取检索费用。
- 对已还原的存档对象调用的后续还原请求将作为 GET 请求计费。有关定价的信息，请参阅 Amazon S3 定价。

恢复已归档的对象

您可以使用 Amazon S3 控制台、Amazon S3 REST API、AWS SDK、AWS Command Line Interface（AWS CLI）或 S3 批量操作还原归档的对象。

使用 S3 控制台

使用 Amazon S3 控制台还原对象

使用以下过程还原已归档到 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类或 S3 Intelligent-Tiering 归档访问或深度归档访问存储层的对象。

还原已归档的对象

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（桶）列表中，选择包含您想要还原的对象的桶的名称。
4. 在 Objects（对象）列表中，选择要还原的一个或多个对象，再选择 Actions（操作），然后选择 Initiate restore（启动还原）。
5. 如果要从 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 进行还原，请在还原的副本可用的天数框中输入您希望归档数据可供访问的天数。
6. 对于检索层，执行以下操作之一：

   - 选择批量检索或标准检索，然后选择启动还原。
   - 选择 Expedited retrieval（加急检索）（仅适用于 S3 Glacier Flexible Retrieval 或 S3 Intelligent-Tiering 归档访问）。如果您要还原 S3 Glacier Flexible Retrieval 中的对象，则可以

选择是否要购买预调配容量来进行加速检索。如果要购买预调配容量，请继续执行下一步。否则，请选择启动还原。

> **Note**
>
> S3 Intelligent-Tiering 归档访问层和深度归档访问层中的对象会自动还原到频繁访问层。

7. （可选）如果您要还原 S3 Glacier Flexible Retrieval 中的对象，并且您选择了加速检索，您可以选择是否购买预调配容量。预调配容量仅适用于 S3 Glacier Flexible Retrieval 中的对象。如果您已有预调配容量，请选择 启动还原以开始预调配检索。

   如果您有预调配容量，则您的预调配容量可处理您的所有加速检索。有关更多信息，请参阅 预调配容量。

   - 如果您没有预调配容量并且不想购买，请选择启动还原。
   - 如果您没有预调配容量，但要购买预调配容量单位（PCU），请选择购买 PCU。在购买 PCU 对话框中，选择要购买的 PCU 数量，确认购买，然后选择购买 PCU。在收到购买成功消息后，选择启动还原以开始预调配检索。

使用 AWS CLI

从 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 还原对象

以下示例使用 `restore-object` 命令还原桶 *amzn-s3-demo-bucket* 中的对象 *dir1/example.obj*，时间为 25 天。

```
aws s3api restore-object --bucket amzn-s3-demo-bucket --key dir1/example.obj --restore-request '{"Days":25,"GlacierJobParameters":{"Tier":"Standard"}}'
```

如果示例中使用的 JSON 句法导致 Windows 客户端错误，则使用如下句法替换还原请求：

```
--restore-request Days=25,GlacierJobParameters={"Tier"="Standard"}
```

从 S3 Intelligent-Tiering 归档访问和深度归档访问还原对象

以下示例使用 `restore-object` 命令将桶 *amzn-s3-demo-bucket* 中的对象 *dir1/example.obj* 还原到频繁访问层

```
aws s3api restore-object --bucket amzn-s3-demo-bucket --key dir1/example.obj --restore-request '{}'
```

> **Note**
>
> 与 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类不同，S3 Intelligent-Tiering 对象的还原请求不接受该 `Days` 值。

监控还原状态

要监控 `restore-object` 请求的状态，请使用以下 `head-object` 命令：

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key dir1/example.obj
```

有关更多信息，请参阅 AWS CLI 命令参考 中的 restore-object。

使用 REST API

Amazon S3 向您提供了用于启动归档对象还原的 API 操作。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 RestoreObject。

使用 AWS SDK

有关如何使用 AWS SDK 还原 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 中的归档对象的示例，请参阅将 `RestoreObject` 与 AWS SDK 或 CLI 配合使用。

使用 S3 批量操作

要使用单个请求还原多个归档对象，您可以使用 S3 批量操作。您为 S3 批量操作提供要操作的对象列表。S3 批量操作调用相应的 API 操作来执行指定的操作。单个批量操作任务可对包含 EB 级数据的数十亿个对象执行指定操作。

要创建批量操作任务，您必须有一个仅包含要还原的对象的清单。您可以使用 S3 清单来创建清单，也可以提供包含必要信息的 CSV 文件。有关更多信息，请参阅 the section called “指定清单”。

在创建和运行 S3 批量操作任务之前，您必须向 Amazon S3 授予代表您执行 S3 批量操作的权限。有关所需的权限，请参阅the section called “授予权限”。

> **Note**
>
> 批量操作任务既可以在 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类对象上运行，也可以在 S3 Intelligent-Tiering 归档访问和深度归档访问存储层对象上运行。批量操作不能在同一个任务中对两种类型的归档对象进行操作。要还原两种类型的对象，必须创建单独的分批操作任务。
>
> 有关使用批量操作还原归档对象的更多信息，请参阅the section called “还原对象”。

创建 S3 启动还原对象批量操作任务

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择批量操作。
3. 选择创建任务。
4. 对于 AWS 区域，选择要在其中创建任务的区域。
5. 在清单格式下，请选择要使用的清单类型。
   - 如果您选择 S3 清单报告，请输入 Amazon S3 作为 CSV 格式清单报告的一部分生成的 `manifest.json` 对象的路径。如果要使用除最新版本之外的清单版本，请输入 `manifest.json` 对象的版本 ID。
   - 如果您选择 CSV，请输入 CSV 格式清单对象的路径。清单对象必须遵循控制台中描述的格式。如果要使用除最新版本之外的版本，则可以选择包含清单对象的版本 ID。
6. 选择下一步。
7. 在操作部分，选择还原。
8. 在还原部分，对于还原源，选择 Glacier Flexible Retrieval 或 Glacier Deep Archive 或 Intelligent-Tiering 归档访问层或深度归档访问层。

   如果您选择 Glacier Flexible Retrieval 或 Glacier Deep Archive，则为还原的副本可用的天数输入一个数字。

   对于检索层，选择要使用的层。
9. 选择下一步。
10. 在配置其他选项页面上，填写以下部分：

- 在其他选项部分，提供任务的描述并指定该任务的优先级编号。编号越大，优先级越高。有关更多信息，请参阅 the section called “分配任务优先级”。
- 在完成报告部分中，选择批量操作是否应创建完成报告。有关完成报告的更多信息，请参阅the section called “完成报告”。
- 在权限部分，您必须向 Amazon S3 授予代表您执行批量操作的权限。有关所需的权限，请参阅the section called “授予权限”。
- （可选）在任务标签部分，以键值对方式添加标签。有关更多信息，请参阅 the section called “使用标签”。

完成后，选择下一步。

11. 在 Review (审核) 页面上，验证设置。如果需要进行更改，请选择 Previous。否则，请选择创建任务。

有关批量操作的更多信息，请参阅使用批量操作还原对象和创建 S3 批量操作任务。

检查还原状态和到期日期

您可以使用 Amazon S3 控制台、Amazon S3 事件通知、AWS CLI 或 Amazon S3 REST API 查看还原请求的状态或到期日期。

> **Note**
>
> 从 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类还原的对象仅存储您指定的天数。以下过程将返回这些副本的到期日期。
> 从 S3 Intelligent-Tiering 存档访问和深度存档访问存储层还原的对象没有到期日期，而是移回到频繁访问层。

使用 S3 控制台

在 Amazon S3 控制台中检查对象的还原状态和过期日期

1. 通过以下网址打开 Simple Storage Service（Amazon S3）控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在桶列表中，选择包含待还原对象的桶的名称。

4. 在对象列表中，选择要还原的对象。此时将显示对象的详细信息页面。

   - 如果还原未完成，则在页面顶部，您会看到一个部分显示正在还原。
   - 如果还原已完成，则在页面顶部，您会看到一个部分显示还原完成。如果您从 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 中进行还原，则此部分还会显示还原过期日期。在该日期，Amazon S3 将删除归档对象的已还原副本。

使用 Amazon S3 事件通知

通过使用 `s3:ObjectRestore:Completed` 操作与 Amazon S3 事件通知特征，您可以收到对象还原完成的通知。有关启用事件通知的更多信息，请参阅 Enabling notifications by using Amazon SQS, Amazon SNS, and AWS Lambda。有关各种 `ObjectRestore` 事件类型的更多信息，请参阅the section called “SQS、SNS 和 Lambda 支持的事件类型”。

使用 AWS CLI

使用 AWS CLI 检查对象的还原状态和过期日期

以下示例使用 `head-object` 命令查看桶 *amzn-s3-demo-bucket* 中对象 *dir1/example.obj* 的元数据。当您对正在还原的对象运行此命令时，Amazon S3 会返回还原是否正在进行以及（如果适用）过期日期。

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key dir1/example.obj
```

预期输出（还原正在进行中）：

```
{
    "Restore": "ongoing-request=\"true\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

预期输出（还原已完成）：

```
{
    "Restore": "ongoing-request=\"false\", expiry-date=\"Wed, 12 Aug 2020 00:00:00 GMT
\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

有关 `head-object` 的更多信息，请参阅《AWS CLI 命令参考》中的 head-object。

使用 REST API

Amazon S3 提供了一个 API 操作供您检索对象元数据。要使用 REST API 检查归档对象的还原状态和过期日期，请参阅《Amazon Simple Storage Service API 参考》中的 HeadObject。

升级正在进行的还原的速度

在还原过程中，您可以升级还原的速度。

将正在进行的还原升级到更快的层级

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在 Buckets（桶）列表中，选择包含您想要还原的对象的桶的名称。
4. 在对象列表中，选择要还原的对象。此时将显示对象的详细信息页面。在对象的详细信息页面上，选择升级检索层。有关检查对象还原状态的信息，请参阅检查还原状态和到期日期。
5. 选择要升级到的层，然后选择启动还原。

使用 S3 对象锁定

S3 对象锁定可以协助在固定的时间长度内或无限期地阻止删除或覆盖 Amazon S3 对象。对象锁定使用一次写入多次读取（WORM）模式存储对象。您可以使用对象锁定来协助满足需要 WORM 存储的法规要求，或用于添加额外的保护层来防止对象被更改和删除。

> **Note**
>
> S3 对象锁定已由 Cohasset Associates 评估，可在受 SEC 17a-4、CFTC 和 FINRA 法规约束的环境中使用。有关对象锁定与这些法规关系的更多信息，请参阅 Cohasset Associates 合规性评估。

对象锁定提供了以下两种方式来管理对象保留：保留期限和依法保留。对象版本可以同时具有保留期限和/或依法保留。

- 保留期限 – 保留期限指定对象可以保持锁定状态的固定时间长度。您可以为各个对象设置唯一的保留期。此外，您可以在 S3 存储桶上设置默认保留期。您还可以在桶策略中使用 `s3:object-lock-remaining-retention-days` 条件键限制允许的最小和最大保留期。这有助于您确定保留期的范围，并限制可能短于或长于该范围的保留期。
- 依法保留 – 依法保留提供的保护与保留期限相同，但没有到期日期。依法保留将一直有效，直至您明确将其删除。法定保留与保留期无关，且应该对单独的对象目标应用。

对象锁定仅在启用 S3 版本控制的存储桶中生效。当您锁定某一对象版本时，Amazon S3 会将锁定信息存储在该对象版本的元数据中。在对象上实施保留期限或依法保留仅保护在请求中指定的版本。保留期和法定保留不会阻止创建对象的新版本，也不会阻止在对象顶部添加删除标记。有关 S3 版本控制的信息，请参阅在 S3 存储桶中使用版本控制。

如果您将一个对象放入存储桶中，而存储桶中包含的现有受保护对象与其具有相同的对象键名，Amazon S3 将创建对象的新版本。现有受保护版本的对象将根据其保留配置保持锁定状态。

S3 对象锁定的工作原理

主题

保留期限

保留期限可在固定时间内保护对象版本。当您对对象版本施加保留期限时，Amazon S3 会在该对象版本的元数据中存储时间戳，以指示保留期限的到期时间。在保留期限到期后，便可覆盖或删除对象版本。

您可以对单独对象版本或在存储桶属性上明确设置保留期限，以便它自动应用于存储桶中的所有对象。将保留期显式应用于对象版本时，即为对象版本指定保留到期日期。Amazon S3 将此日期存储在对象版本的元数据中。

您还可以在存储桶的属性中设置保留期限。在存储存储桶上设置保留期限时，您可以指定放入存储桶中的每个对象版本受保护的持续时间，以天或年为单位。当您将对象放入存储桶时，Amazon S3 将该对象版本的创建时间戳加上指定的持续时间，以此来计算保留到期日。然后，在您对该对象版本明确应用的单独锁定保留期限内，对象版本将受到保护。

> **Note**
>
> 当您在存储桶中 PUT 具有明确的单独保留模式和期限的对象版本时，该对象版本的单独对象锁定设置会覆盖所有存储桶属性保留设置。

与所有其他对象锁定设置一样，保留期限适用于各个对象版本。单个对象的不同版本可以具有不同的保留模式和周期。

例如，如果您将 15 天的对象放入 30 天保留期限，并将一个保留期限为 60 天的同名对象 PUT 到 Amazon S3 中。在这种情况下，您的 PUT 请求将成功，Amazon S3 将创建一个保留期限为 60 天的新对象版本。较旧版本将保持其原来的保留期限，并且可在 15 天内删除。

在某个对象版本上应用了保留设置后，您可以延长保留期限。要执行此操作，您需对该对象版本提交新对象锁定请求，其保留到期日晚于当前为该对象版本配置的保留到期日期。Amazon S3 将用更长的新期限替换现有的保留期限。任何用户只要具有实施对象保留期限的权限，即可对象版本的保留期限。要设置保留期限，您必须具有 `s3:PutObjectRetention` 权限。

在对象或 S3 存储桶上设置保留权期限时，您必须选择两种保留模式之一：合规性或监管。

保留模式

S3 对象锁定提供了两种保留模式，可为对象应用不同级别的保护：

- 合规性模式

- 监管模式

在合规性模式中，任何用户都不能覆盖或删除受保护的对象版本，包括 AWS 账户中的根用户。在合规性模式下锁定对象后，其保留模式便无法更改，其保留期限也不能缩短。合规性模式可帮助确保在保留期限内无法覆盖或删除对象版本。

> **Note**
>
> 在合规模式下，要想在保留日期到期前删除对象，唯一的方法就是删除相关的 AWS 账户。

在监管 模式中，除非用户具有特殊权限，否则用户不能覆盖或删除对象版本，或更改其锁定设置。使用监管模式，您可以保护对象以免被大多数用户删除，但您仍可向部分用户授予权限，以在必要时更改保留设置或删除对象。您还可以在创建合规性模式保留期限之前使用监管模式测试保留期限设置。

要覆盖或删除监管模式保留设置，您必须具有 `s3:BypassGovernanceRetention` 权限，并且在需要覆盖监管模式的任何请求中，必须明确包含 `x-amz-bypass-governance-retention:true` 作为请求标头。

> **Note**
>
> 默认情况下，Amazon S3 控制台包括 `x-amz-bypass-governance-retention:true` 标头。如果您尝试删除受监管模式保护的对象并且拥有 `s3:BypassGovernanceRetention` 权限，则操作将会成功。

依法保留

使用对象锁定，您还可以在对象版本上实施依法保留。与保留期限相似，依法保留可防止对象版本被覆盖或删除。但是，依法保留没有关联的固定时间长度，会一直有效，直至删除。拥有 `s3:PutObjectLegalHold` 权限的任何用户均可自由实施和删除依法保留。

依法保留与保留期限无关。对对象版本实施依法保留不会影响该对象版本的保留模式或保留期限。

例如，假设您对某一对象版本实施依法保留，同时该对象版本也受保留期限保护。如果保留期限过期，该对象不会失去其 WORM 保护。那么依法保留将继续保护该对象，直至授权用户明确删除依法保留对

象。与之相似，如果您在对象版本的保留期限生效时删除依法保留，该对象版本将一直受保护，直至保留期限过期。

使用 S3 对象锁定的最佳实践

如果您想保护对象在预定义的保留期内不被大多数用户删除，但同时又希望一些具有特殊权限的用户能够灵活地更改保留设置或删除对象，请考虑使用监管模式。

如果您从不希望任何用户（包括您的 AWS 账户中的根用户）能够在预定义的保留期内删除对象，请考虑使用合规模式。如果您需要存储合规数据，则可以使用此模式。

当您不确定要让对象保持不可变的时长时，可以使用依法保留。这可能是因为您即将对数据进行外部审计，并希望在审计完成之前保持对象不可变。或者，您可能有一个正在进行的项目，而您希望在项目完成之前，让该项目使用的数据集保持不变。

所需的权限

对象锁定操作需要特定的权限。根据您正在尝试的确切操作，您可能需要以下任意权限：

- `s3:BypassGovernanceRetention`
- `s3:GetBucketObjectLockConfiguration`
- `s3:GetObjectLegalHold`
- `s3:GetObjectRetention`
- `s3:PutBucketObjectLockConfiguration`
- `s3:PutObjectLegalHold`
- `s3:PutObjectRetention`

有关 Amazon S3 权限的完整列表及其描述，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

有关将条件与权限结合使用的信息，请参阅 使用条件键的存储桶策略示例。

对象锁定注意事项

Amazon S3 对象锁定可以协助在固定的时间长度内或无限期地阻止删除或覆盖对象。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API 来查看或设置对象锁定信息。有关 S3 对象锁定功能的一般信息，请参阅使用 S3 对象锁定。

> **Important**
>
> - 在某个存储桶上启用对象锁定后，您将无法为该存储桶禁用对象锁定或暂停版本控制。
> - 具有对象锁定的 S3 存储桶不能用作服务器访问日志的目标存储桶。有关更多信息，请参阅 the section called “记录服务器访问”。

主题

查看锁定信息的权限

您可以使用 HeadObject 或 GetObject 操作，以编程方式查看 Amazon S3 对象版本的对象锁定状态。这两个操作都会返回指定对象版本的保留模式、保留到期日和依法保留状态。此外，您可以使用 S3 清单来查看 S3 存储桶中多个对象的对象锁定状态。

要查看对象版本的保留模式和保留期限，您必须拥有 `s3:GetObjectRetention` 权限。要查看对象版本的依法保留状态，您必须拥有 `s3:GetObjectLegalHold` 权限。要查看存储桶的默认保留配置，您必须具有 `s3:GetBucketObjectLockConfiguration` 权限。如果您对未启用 S3 对象锁定的存储桶请求对象锁定配置，则 Amazon S3 将返回错误。

绕过监管模式

如果您具有 `s3:BypassGovernanceRetention` 权限，则在对象未受保护时，您可对在监管模式下锁定的对象版本执行操作。这些操作包括删除对象版本、缩短保留期限，或者通过实施参数为空的新 `PutObjectRetention` 请求来删除对象锁定保留期限。

要绕过监管模式，您必须在请求中明确指明您要绕过此模式。为此，在 PutObjectRetention API 操作请求中包含 x-amz-bypass-governance-retention:true 标头，或将等效参数用于通过 AWS CLI 或 AWS SDK 发出的请求。如果您具有 s3:BypassGovernanceRetention 权限，则 S3 控制台会自动将此标头应用于通过 S3 控制台发出的请求。

> **Note**
>
> 绕过监管模式不会影响对象版本的依法保留状态。如果某个对象版本已启用依法保留，则依法保留将保持有效，并可阻止覆盖或删除该对象版本的请求。

将对象锁定与 S3 复制结合使用

您可以将对象锁定与 S3 复制结合使用，以便跨 S3 存储桶，对锁定对象及其保留元数据启用自动异步复制。这意味着，对于复制的对象，Amazon S3 采用源存储桶的对象锁定配置。换句话说，如果源存储桶已启用对象锁定，则目标存储桶也必须启用对象锁定。如果将对象直接上传到目标存储桶（在 S3 复制之外），它将使用目标存储桶上设置的对象锁定。如果您使用复制功能，则源存储桶中的对象会被复制到目标存储桶。

要在启用了对象锁定的存储桶上设置复制功能，您可以使用 S3 控制台、AWS CLI、Amazon S3 REST API 或 AWS SDK。

> **Note**
>
> 要将对象锁定与复制功能结合使用，您必须在用于设置复制的 AWS Identity and Access Management（IAM）角色中，授予对源 S3 存储桶的两项额外的权限。这两项额外的权限是 s3:GetObjectRetention 和 s3:GetObjectLegalHold。如果角色已有 s3:Get* 权限声明，则该声明已满足要求。有关更多信息，请参阅 为实时复制设置权限。
> 有关 S3 复制的一般信息，请参阅复制对象概述。
> 有关设置 S3 复制的示例，请参阅配置实时复制的示例。

将对象锁定与 Amazon S3 清单结合使用

您可以配置 Amazon S3 清单，以便按预定的计划创建 S3 存储桶中对象的列表。对于您的对象，您可以将 Amazon S3 清单配置为包含以下对象锁定元数据：

- 保留截止日
- 保留模式

- 依法保留状态

有关更多信息，请参阅 Amazon S3 清单。

使用对象锁定管理 S3 生命周期策略

对象生命周期管理配置将继续正常应用于受保护的对象，包括实施删除标记。但是，S3 生命周期过期策略无法删除对象的锁定版本。无论对象位于哪个存储类，以及在存储类之间的整个 S3 生命周期转换过程中，都将保持对象锁定。

有关管理对象生命周期的更多信息，请参阅管理存储生命周期。

使用对象锁定管理删除标记

尽管您无法删除受保护的对象版本，但您仍可为该对象创建删除标记。对对象实施删除标记不会删除对象或它的对象版本。但会使 Amazon S3 的行为方式在最大程度上与对象被删除时相似。有关更多信息，请参阅 使用删除标记。

> **Note**
>
> 无论基础对象中实施任何保留期限或依法保留，删除标记均不受 WORM 保护。

将 S3 Storage Lens 存储分析功能与对象锁定结合

要查看启用对象锁定的存储字节数和对象计数的指标，您可以使用 Amazon S3 Storage Lens 存储统计管理工具。S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。

有关更多信息，请参阅 使用 S3 Storage Lens 存储统计管理工具保护您的数据。

要获得指标的完整列表，请参阅 Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

将对象上传到启用了对象锁定的桶

如果使用对象锁定为某个对象配置了保留期，则任何上传该对象的请求都需要 `Content-MD5` 标头。MD5 摘要是一种在将对象上传到存储桶后验证对象完整性的方法。上传对象后，Amazon S3 会计算对象的 MD5 摘要，并将其与您提供的值进行比较。只有在两个摘要匹配的情况下，请求才会成功。S3 控制台会自动添加此标头，但您在使用 PutObject API 时必须指定此标头。

有关更多信息，请参阅 在上传对象时使用 Content-M5。

配置事件和通知

您可以使用 Amazon S3 事件通知，通过 AWS CloudTrail 跟踪在对象锁定配置及数据上进行的访问和更改。有关 CloudTrail 的信息，请参阅《AWS CloudTrail 用户指南》中的什么是 AWS CloudTrail？

您还可以使用 Amazon CloudWatch 根据此数据生成警报。有关 CloudWatch 的信息，请参阅《Amazon CloudWatch 用户指南》中的什么是 Amazon CloudWatch？

使用存储桶策略为保留期限设置限制

您可以使用存储桶策略为存储桶设置允许的最小和最大保留期限。最长保留周期为 100 年。

以下示例显示一个存储桶策略，它使用 `s3:object-lock-remaining-retention-days` 条件键将最长保留期限设置为 10 天。

```
{
    "Version": "2012-10-17",
    "Id": "SetRetentionLimits",
    "Statement": [
        {
            "Sid": "SetRetentionPeriod",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "NumericGreaterThan": {
                    "s3:object-lock-remaining-retention-days": "10"
                }
            }
        }
    ]
}
```

> **Note**
>
> 如果您的存储桶是复制配置的目标存储桶，则对于通过使用复制功能创建的对象副本，您可以设置允许的最小和最大保留期限。为此，您必须在存储桶策略中允许 `s3:ReplicateObject` 操作。有关复制权限的更多信息，请参阅the section called “设置权限”。

有关存储桶的更多信息，请参阅以下主题：

- 《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。
- 对象操作
- 使用条件键的存储桶策略示例

配置 S3 对象锁定

借助 Amazon S3 对象锁定，您可以使用一次写入多次读取（WORM）模式在 Amazon S3 中存储对象。您可以使用 S3 对象锁定在固定的时间段内或无限期地阻止删除或覆盖对象。有关对象锁定功能的一般信息，请参阅使用 S3 对象锁定。

在锁定任何对象之前，必须在存储桶上启用 S3 版本控制和对象锁定。之后，您可以设置保留期和/或依法保留。

要使用对象锁定功能，您必须拥有一定的权限。有关与各种对象锁定操作相关的权限列表，请参阅the section called “所需的权限”。

> **Important**
>
> - 在某个存储桶上启用对象锁定后，您将无法为该存储桶禁用对象锁定或暂停版本控制。
> - 具有对象锁定的 S3 存储桶不能用作服务器访问日志的目标存储桶。有关更多信息，请参阅 the section called “记录服务器访问”。

主题

- 创建新的 S3 存储桶时启用对象锁定
- 在现有 S3 存储桶上启用对象锁定
- 设置或修改 S3 对象的依法保留
- 设置或修改 S3 对象的保留期
- 设置或修改 S3 存储桶的默认保留期

创建新的 S3 存储桶时启用对象锁定

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API 在创建新的 S3 存储桶时启用对象锁定。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择创建存储桶。

   此时将打开创建存储桶页面。
4. 对于存储桶名称，输入存储桶的名称。

   > **Note**
   >
   > 创建存储桶后，便无法更改其名称。有关给存储桶命名的更多信息，请参阅存储桶命名规则。

5. 对于区域，选择要放置存储桶的 AWS 区域。
6. 在对象所有权下，选择禁用或启用访问控制列表（ACL），并控制上传到存储桶中的对象的所有权。
7. 在此存储桶的屏蔽公共访问权限设置下，选择要应用于存储桶的屏蔽公共访问权限设置。
8. 在存储桶版本控制下，选择已启用。

   对象锁定仅适用于受版本控制的存储桶。
9. （可选）在标签下，您可以选择向存储桶添加标签。标签是用于对存储进行分类和分配成本的键/值对。
10. 在高级设置下，找到对象锁定，然后选择启用。

    您必须确认，启用对象锁定将永久允许锁定此存储桶中的对象。
11. 选择创建存储桶。

使用 AWS CLI

以下 `create-bucket` 示例创建了一个名为 *`amzn-s3-demo-bucket1`* 的新 S3 存储桶，该存储桶启用了对象锁定：

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket1 --object-lock-enabled-for-bucket
```

有关更多信息和示例，请参阅《AWS CLI 命令参考》中的 create-bucket。

> **Note**
>
> 您可以从控制台中使用 AWS CloudShell 运行 AWS CLI 命令。AWS CloudShell 是一个基于浏览器、经过预验证的 Shell，您可以直接从AWS Management Console中启动它。有关更多信息，请参阅《AWS CloudShell 用户指南》中的什么是 CloudShell？

使用 REST API

您可以使用 REST API 创建一个启用了对象锁定的新 S3 存储桶。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CreateBucket。

使用 AWS SDK

有关在使用 AWS SDK 创建新 S3 存储桶时如何启用对象锁定的示例，请参阅将 `CreateBucket` 与 AWS SDK 或 CLI 配合使用。

有关如何使用 AWS SDK 获取当前对象锁定配置的示例，请参阅将 `GetObjectLockConfiguration` 与 AWS SDK 或 CLI 配合使用。

有关演示使用 AWS SDK 的不同对象锁定功能的交互式场景，请参阅通过 AWS SDK 使用 Amazon S3 对象锁定功能。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

在现有 S3 存储桶上启用对象锁定

您可以使用 Amazon S3 控制台、AWS CLI、AWS SDK 或 Amazon S3 REST API 为现有 S3 存储桶启用对象锁定。

使用 S3 控制台

> **Note**
>
> 对象锁定仅适用于受版本控制的存储桶。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。

3. 在存储桶列表中，选择要为其启用对象锁定的存储桶的名称。
4. 选择属性选项卡。
5. 在属性下，向下滚动到对象锁定部分，然后选择编辑。
6. 在对象锁定下，选择启用。

   您必须确认，启用对象锁定将永久允许锁定此存储桶中的对象。
7. 选择 Save changes（保存更改）。

使用 AWS CLI

以下 `put-object-lock-configuration` 示例命令在名为 *amzn-s3-demo-bucket1* 的存储桶上设置 50 天的对象锁定保留期：

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-
lock-configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

有关更多信息和示例，请参阅《AWS CLI 命令参考》中的 put-object-lock-configuration。

> **Note**
>
> 您可以从控制台中使用 AWS CloudShell 运行 AWS CLI 命令。AWS CloudShell 是一个基于浏览器、经过预验证的 Shell，您可以直接从AWS Management Console中启动它。有关更多信息，请参阅《AWS CloudShell 用户指南》中的什么是 CloudShell？

使用 REST API

您可以使用 Amazon S3 REST API 在现有 S3 存储桶上启用对象锁定。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutObjectLockConfiguration。

使用 AWS SDK

有关如何使用 AWS SDK 为现有 S3 存储桶启用对象锁定的示例，请参阅将 PutObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用。

有关如何使用 AWS SDK 获取当前对象锁定配置的示例，请参阅将 GetObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用。

有关演示使用 AWS SDK 的不同对象锁定功能的交互式场景，请参阅通过 AWS SDK 使用 Amazon S3 对象锁定功能。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

设置或修改 S3 对象的依法保留

您可以使用 Amazon S3 控制台、AWS CLI、AWS SDK 或 Amazon S3 REST API 设置或移除 S3 对象的依法保留。

> **Important**
>
> - 如果要对一个对象设置依法保留，则该对象的存储桶必须已启用对象锁定。
> - 当您在存储桶中 PUT 具有明确的单独保留模式和期限的对象版本时，该对象版本的单独对象锁定设置会覆盖所有存储桶属性保留设置。

有关更多信息，请参阅 the section called “依法保留”。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择包含要设置或修改其依法保留的对象的存储桶名称。
4. 在对象列表中，选择要设置或修改其依法保留的对象。
5. 在对象属性页面上，找到对象锁定依法保留部分，然后选择编辑。
6. 选择启用以设置依法保留，或选择禁用以移除依法保留。
7. 选择 Save changes（保存更改）。

使用 AWS CLI

以下 `put-object-legal-hold` 示例为名为 *amzn-s3-demo-bucket1* 的存储桶中的对象 *my-image.fs* 设置了依法保留：

```
aws s3api put-object-legal-hold --bucket amzn-s3-demo-bucket1 --key my-image.fs --
legal-hold="Status=ON"
```

以下 put-object-legal-hold 示例为名为 *amzn-s3-demo-bucket1* 的存储桶中的对象 *my-image.fs* 移除了依法保留：

```
aws s3api put-object-legal-hold --bucket amzn-s3-demo-bucket1 --key my-image.fs --
legal-hold="Status=OFF"
```

有关更多信息和示例，请参阅《AWS CLI 命令参考》中的 put-object-legal-hold。

> **Note**
>
> 您可以从控制台中使用 AWS CloudShell 运行 AWS CLI 命令。AWS CloudShell 是一个基于浏览器、经过预验证的 Shell，您可以直接从AWS Management Console中启动它。有关更多信息，请参阅《AWS CloudShell 用户指南》中的什么是 CloudShell？

使用 REST API

您可以使用 REST API 设置或修改对象的依法保留。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutObjectLegalHold。

使用 AWS SDK

有关如何使用 AWS SDK 设置对象法定保留的示例，请参阅将 PutObjectLegalHold 与 AWS SDK 或 CLI 配合使用。

有关如何使用 AWS SDK 获取当前法定保留状态的示例，请参阅使用 AWS SDK 获取 Amazon S3 存储桶的法定保留配置。

有关演示使用 AWS SDK 的不同对象锁定功能的交互式场景，请参阅通过 AWS SDK 使用 Amazon S3 对象锁定功能。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

设置或修改 S3 对象的保留期

您可以使用 Amazon S3 控制台、AWS CLI、AWS SDK 或 Amazon S3 REST API 设置或修改 S3 对象的保留期。

> **Important**
>
> - 如果要对一个对象设置保留期，则该对象的存储桶必须已启用对象锁定。

- 当您在存储桶中 PUT 具有明确的单独保留模式和期限的对象版本时，该对象版本的单独对象锁定设置会覆盖所有存储桶属性保留设置。
- 在合规模式下，要想在保留日期到期前删除对象，唯一的方法就是删除相关的 AWS 账户。

有关更多信息，请参阅 保留期限。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择包含要设置或修改其保留期的对象的存储桶名称。
4. 在对象列表中，选择要设置或修改其保留期的对象。
5. 在对象属性页面上，找到对象锁定保留部分，然后选择编辑。
6. 在保留下，选择启用以设置保留期，或者选择禁用以删除保留期。
7. 如果选择启用，请在保留模式下选择治理模式或合规模式。有关更多信息，请参阅 保留模式。
8. 在保留截止日期下，选择您希望保留期结束的日期。在该时间段内，您的对象将受 WORM 保护，不能被覆盖或删除。有关更多信息，请参阅 保留期限。
9. 选择 Save changes（保存更改）。

使用 AWS CLI

以下 `put-object-retention` 示例将名为 *amzn-s3-demo-bucket1* 的存储桶中对象 *my-image.fs* 的保留期限设置为 2025 年 1 月 1 日：

```
aws s3api put-object-retention --bucket amzn-s3-demo-bucket1 --key my-image.fs --
retention='{ "Mode": "GOVERNANCE", "RetainUntilDate": "2025-01-01T00:00:00" }'
```

有关更多信息和示例，请参阅《AWS CLI 命令参考》中的 put-object-retention。

> **Note**
>
> 您可以从控制台中使用 AWS CloudShell 运行 AWS CLI 命令。AWS CloudShell 是一个基于浏览器、经过预验证的 Shell，您可以直接从AWS Management Console中启动它。有关更多信息，请参阅《AWS CloudShell 用户指南》中的什么是 CloudShell？

使用 REST API

您可以使用 REST API 设置对象的保留期。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutObjectRetention。

使用 AWS SDK

有关如何使用 AWS SDK 为对象设置保留期的示例，请参阅将 PutObjectRetention 与 AWS SDK 或 CLI 配合使用。

有关如何使用 AWS SDK 获取对象保留期的示例，请参阅将 GetObjectRetention 与 AWS SDK 或 CLI 配合使用。

有关演示使用 AWS SDK 的不同对象锁定功能的交互式场景，请参阅通过 AWS SDK 使用 Amazon S3 对象锁定功能。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

设置或修改 S3 存储桶的默认保留期

您可以使用 Amazon S3 控制台、AWS CLI、AWS SDK 或 Amazon S3 REST API 设置或修改 S3 存储桶的默认保留期。您可以指定放入存储桶中的每个对象版本受保护的持续时间，以天或年为单位。

Important

- 如果要为某个存储桶设置默认保留期，则该存储桶必须已启用对象锁定。
- 当您在存储桶中 PUT 具有明确的单独保留模式和期限的对象版本时，该对象版本的单独对象锁定设置会覆盖所有存储桶属性保留设置。
- 在合规模式下，要想在保留日期到期前删除对象，唯一的方法就是删除相关的 AWS 账户。

有关更多信息，请参阅 保留期限。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。

3. 在存储桶列表中，选择要设置或修改其默认保留期的存储桶的名称。
4. 选择属性选项卡。
5. 在属性下，向下滚动到对象锁定部分，然后选择编辑。
6. 在默认保留期下，选择启用以设置默认保留期，或者选择禁用以删除默认保留期。
7. 如果选择启用，请在保留模式下选择治理模式或合规模式。有关更多信息，请参阅 保留模式。
8. 在默认保留期下，选择您希望保留期持续的天数或年数。放置在此存储桶中的对象将在此天数或年数内被锁定。有关更多信息，请参阅 保留期限。
9. 选择 Save changes（保存更改）。

使用 AWS CLI

以下 `put-object-lock-configuration` 示例命令使用合规模式在名为 *amzn-s3-demo-bucket1* 的存储桶上设置 50 天的对象锁定保留期：

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-
lock-configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

以下 `put-object-lock-configuration` 示例删除了存储桶上的默认保留配置：

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled"}'
```

有关更多信息和示例，请参阅《AWS CLI 命令参考》中的 put-object-lock-configuration。

> **Note**
>
> 您可以从控制台中使用 AWS CloudShell 运行 AWS CLI 命令。AWS CloudShell 是一个基于浏览器、经过预验证的 Shell，您可以直接从AWS Management Console中启动它。有关更多信息，请参阅《AWS CloudShell 用户指南》中的什么是 CloudShell？

使用 REST API

您可以使用 REST API 在现有的 S3 存储桶上设置默认保留期。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutObjectLockConfiguration。

使用 AWS SDK

有关如何使用 AWS SDK 在现有 S3 存储桶上设置默认保留期的示例，请参阅将 PutObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用。

有关演示使用 AWS SDK 的不同对象锁定功能的交互式场景，请参阅通过 AWS SDK 使用 Amazon S3 对象锁定功能。

有关使用不同 AWS SDK 的一般信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。

使用 Amazon S3 存储类

Amazon S3 中的每个对象都有与之关联的存储类。例如，如果您列出 S3 存储桶中的对象，则控制台会在列表中显示所有对象的存储类。Amazon S3 将为您存储的对象提供一系列存储类。请根据您的使用案例场景和性能访问要求选择一个类。所有这些存储类都提供高持久性存储。

以下各节提供了关于各种存储类以及如何为对象设置存储类的详细信息。

主题

经常访问对象的存储类

对于性能敏感的使用案例（需要毫秒级访问时间的用例）和经常访问的数据，Amazon S3 提供以下存储类：

- S3 Standard – 默认存储类。如果在上传对象时未指定存储类，Amazon S3 会分配 S3 Standard 存储类。
- S3 Express One Zone – Amazon S3 Express One Zone 是高性能的单区 Amazon S3 存储类，专门用于为延迟要求极高的应用程序提供稳定的毫秒级数据访问。S3 Express One Zone 是目前具有极低延迟的云对象存储类，相比 S3 Standard，其数据访问速度要快 10 倍，且请求成本低 50%。使用

S3 Express One Zone，您的数据将冗余地存储在单个可用区中的多个设备上。有关更多信息，请参阅 什么是 S3 Express One Zone？。

- 低冗余存储 – 低冗余存储（RRS）存储类设计用于可使用低于 S3 Standard 存储类的冗余级别存储的非关键性、可复现数据。

> **Important**
>
> 我们建议不要使用此存储类。S3 Standard 存储类更经济高效。

为了实现持久性，RRS 对象的预期平均每年对象损失率为 0.01%。如果 RRS 对象丢失，则在对该对象发出请求时，Amazon S3 会返回 405 错误。

用于自动优化访问模式不断变化或未知的数据的存储类

S3 Intelligent-Tiering 是一种 Amazon S3 存储类，旨在通过自动将数据移动到最具成本效益的访问层来优化存储成本，而不会影响性能或产生运营开销。S3 Intelligent-Tiering 是可实现以下目标的唯一一种云存储类：它通过在访问模式更改时在访问层之间的细粒度对象级别上移动数据，实现自动成本节约。如果您希望针对具有未知或不断变化的访问模式的数据优化存储成本，S3 Intelligent-Tiering 存储类非常适合。S3 Intelligent-Tiering 没有检索费用。

每月只需支付少量的对象监控和自动化费用，S3 Intelligent-Tiering 即可监控访问模式并自动将尚未访问的对象移动到成本较低的访问层。S3 Intelligent-Tiering 可利用三个低延迟和高吞吐量访问层自动节省存储成本。对于可以异步访问的数据，您可以选择激活 S3 Intelligent-Tiering 存储类中的自动存档功能。S3 Intelligent-Tiering 专为 99.9% 的可用性和 99.999999999% 的耐用性而设计。

S3 Intelligent-Tiering 会自动将对象存储在三个访问层：

- 频繁访问 – 上传或转换到 S3 Intelligent-Tiering 的对象将自动存储在频繁访问层中。
- 不频繁访问 – S3 Intelligent-Tiering 将连续 30 天未访问的对象移动到不频繁访问层。
- 归档即时访问 – 使用 S3 Intelligent-Tiering，连续 90 天未访问的任何现有对象都会自动移动到归档即时访问层。

除了这三个层之外，S3 Intelligent-Tiering 还提供两个可选的归档访问层：

- 归档访问 – S3 Intelligent-Tiering 为您提供了一个选项，可让您对可以异步访问的数据激活归档访问层。激活后，归档访问层会自动存档至少连续 90 天未访问的对象。

- 深度归档访问 – S3 Intelligent-Tiering 为您提供了一个选项，可让您对可以异步访问的数据激活深度归档访问层。激活后，深度归档访问层会自动存档至少连续 180 天未访问的对象。

> **Note**
>
> - 如果您想绕过归档即时访问层，则只激活 90 天的归档访问层。归档访问层提供了略低的存储成本，只需几分钟到一小时的检索时间。归档即时访问层提供毫秒级的访问和高吞吐量性能。
> - 仅当您的应用程序可以异步访问对象时，才激活归档访问和深层归档访问层。如果您正在检索的对象存储在归档访问层或深度归档访问层中，请首先使用 `RestoreObject` 还原对象。

您可以将新创建的数据移动到 S3 Intelligent-Tiering，并将其设置为默认存储类。您还可以使用 `PutBucketIntelligentTieringConfiguration` API 操作、AWS CLI 或 Amazon S3 控制台，选择激活其中一个或这两个归档访问层。有关使用 S3 Intelligent-Tiering 和激活归档访问层的更多信息，请参阅 使用 S3 Intelligent-Tiering。

要访问归档访问层或深度归档访问层中的对象，您首先需要将其还原。有关更多信息，请参阅 从 S3 Intelligent-Tiering 归档访问和深度归档访问层恢复对象。

> **Note**
>
> 如果对象大小小于 128 KB，未被监控，不符合自动分层条件。较小的对象始终存储在频繁访问层中。有关 S3 Intelligent-Tiering 的更多信息，请参阅S3 Intelligent-Tiering 访问层。。

不经常访问对象的存储类

S3 Standard-IA 和 S3 One Zone-IA 存储类用于长时间运行且不经常访问的数据。（IA 表示不经常访问。）S3 Standard-IA 和 S3 One Zone-IA 对象可用于毫秒级访问（类似于 S3 Standard存储类）。Amazon S3 会收取这些对象的检索费，因此，它们最适合不经常访问的数据。有关定价信息，请参阅 Amazon S3 定价。

例如，您可以选择 S3 Standard-IA 和 S3 One Zone-IA 存储类执行以下操作：

- 用于存储备份。

- 用于不经常访问但仍需要毫秒级访问的旧数据。例如，上传数据时，您可能会选择 S3 Standard存储类，然后使用生命周期配置指示 Amazon S3 将对象转换为 S3 Standard-IA 或 S3 One Zone-IA 类。

  有关生命周期管理的更多信息，请参阅 管理存储生命周期。

> **Note**
>
> S3 Standard-IA 和 S3 One Zone-IA 存储类适用于您计划存储至少 30 天并且大小超过 128 KB 的对象。如果对象小于 128 KB，Amazon S3 会收取 128 KB 的费用。如果您在 30 天的最短存储期结束前删除对象，则需支付 30 天的费用。在 30 天之前删除、覆盖或转换为其他存储类的对象将产生正常存储使用费，外加最低 30 天剩余时间的按比例收费。有关定价信息，请参阅 Amazon S3 定价。

这些存储类在以下方面有所不同：

- S3 Standard-IA – Amazon S3 跨多个地理位置独立的可用区以冗余方式存储对象数据（类似于 S3 Standard 存储类）。S3 Standard-IA 对象可在出现可用区丢失时复原。此存储类可提供比 S3 One Zone-IA 类更好的可用性和弹性。
- S3 One Zone-IA – Amazon S3 只在一个可用区存储对象数据，因此比 S3 Standard-IA 更便宜。但是，数据无法灵活地应对由于地震和洪水灾害而造成可用区物理丢失的情况。S3 One Zone-IA 存储类和 S3 Standard-IA 一样具有持久性，但是可用性和弹性较差。有关存储类的持久性和可用性比较，请参阅此部分结尾的 比较 Amazon S3 存储类。有关定价信息，请参阅 Amazon S3 定价。

我们建议执行下列操作：

- S3 Standard-IA – 用于主数据或仅无法重新创建的数据副本。
- S3 One Zone-IA – 如果在可用区出现故障时可重新创建数据，可在配置 S3 跨区域复制（CRR）时用于对象副本。

极少访问的对象的存储类

S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类专为低成本、长期数据存储和数据归档而设计。这些存储类提供与 S3 Standard 和 S3 Standard-IA 存储类相同的持久性和弹性。有关 S3 Glacier 存储类的更多信息，请参阅使用 S3 Glacier 存储类作为长期数据存储。

Amazon S3 提供以下 S3 Glacier 存储类：

- S3 Glacier Instant Retrieval – 用于存储极少访问且需要毫秒检索的长期数据。此存储类中的数据可用于实时访问。
- S3 Glacier Flexible Retrieval – 用于可能需要在几分钟内检索其部分数据的归档。此存储类中的数据已归档，不可用于实时访问。
- S3 Glacier Deep Archive – 用于归档很少需要访问的数据。此存储类中的数据已归档，不可用于实时访问。

检索存档的对象

您可以按照与其他存储类相同的方式将对象的存储类设置为 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive，如 设置对象的存储类 部分所述。但是，S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 对象已归档，不可用于实时访问。有关更多信息，请参阅 归档存储。

> **Note**
>
> 如果使用 S3 Glacier 存储类，您的对象将保留在 Amazon S3 中。您无法直接通过单独的 Amazon S3 Glacier 服务访问它们。有关 Amazon S3 Glacier 的更多信息，请参阅 Amazon S3 Glacier 开发人员指南。

Amazon S3 on Outposts 的存储类

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 资源上创建 S3 存储桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序存储和检索对象。您可以像在 Amazon S3 中一样在 AWS Outposts 上使用相同的 API 操作和特征，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS CLI、AWS SDK 或 REST API 使用 S3 on Outposts。

S3 on Outposts 提供了一个新的存储类 S3 Outposts (OUTPOSTS)。S3 Outposts 存储类仅可用于存储在 Outposts 上的存储桶中的对象。如果您尝试将此存储类与 AWS 区域中的 S3 存储桶一起使用，则会出现 `InvalidStorageClass` 错误。此外，如果您尝试将其他 S3 存储类与存储在 S3 on Outposts 存储桶中的对象一起使用，则会出现相同的错误。

存储在 S3 Outposts (OUTPOSTS) 存储类中的对象始终使用具有 Amazon S3 托管式加密密钥的服务器端加密 (SSE-S3) 进行加密。有关更多信息，请参阅 使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

您还可以明确选择使用具有客户提供的加密密钥的服务器端加密 (SSE-C) 来加密存储在 S3 Outposts 存储类中的对象。有关更多信息，请参阅 使用具有客户提供的密钥的服务器端加密（SSE-C）。

> **Note**
>
> S3 on Outposts 不支持带 AWS Key Management Service (AWS KMS) 密钥的服务器端加密 (SSE-KMS)。

有关 S3 on Outposts 的更多信息，请参阅什么是 Amazon S3 on Outposts？

比较 Amazon S3 存储类

下表比较了存储类，包括其可用性、持久性、最短存储持续时间和其他注意事项。

Storage Class	Designed for	Durability (designed for)	Availability (designed for)	Availability Zones	Min storage duration	Min billable object size	Other Considerations
STANDARD	Frequently accessed data	99.999999999%	99.99%	>= 3	None	None	None
STANDARD_IA	Long-lived, infrequently accessed data	99.999999999%	99.9%	>= 3	30 days	128 KB	Per GB retrieval fees apply.
INTELLIGENT_TIERING	Long-lived data with changing or unknown access patterns	99.999999999%	99.9%	>= 3	30 days	None	Monitoring and automation fees per object apply. No retrieval fees.
ONEZONE_IA	Long-lived, infrequently accessed, non-critical data	99.999999999%	99.5%	1	30 days	128 KB	Per GB retrieval fees apply. Not resilient to the loss of the Availability Zone.
GLACIER	Long-term data archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after you restore objects)	>= 3	90 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
DEEP_ARCHIVE	Archiving rarely accessed data with a default retrieval time of 12 hours	99.999999999%	99.99% (after you restore objects)	>= 3	180 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
RRS (Not recommended)	Frequently accessed, non-critical data	99.99%	99.99%	>= 3	None	None	None

* S3 Glacier Flexible Retrieval 要求为每个归档对象提供 40KB 的额外元数据。这包括以 S3 Glacier Flexible Retrieval 费率计费的 32KB 元数据（识别和检索您的数据所需），以及按 S3 Standard 费率计费的额外 8KB 数据。需要使用 S3 Standard 速率来维护归档到 S3 Glacier Flexible Retrieval 的对象的用户定义名称和元数据。有关存储类的更多信息，请参阅 Amazon S3 存储类。

** S3 Glacier Deep Archive 要求为每个归档对象提供 40KB 的额外元数据。这包括以 S3 Glacier Deep Archive 费率计费的 32KB 元数据（识别和检索您的数据所需），以及按 S3 Standard 费率计费的额外 8KB 数据。需要使用 S3 Standard 速率来维护归档到 Amazon S3 Glacier Deep Archive 的对象的用户定义名称和元数据。有关存储类的更多信息，请参阅 Amazon S3 存储类。

请注意，除了 S3 One Zone-IA 和 S3 Express One Zone 之外，所有的存储类都被设计为能够应对灾难导致的可用区的物理损失。除了应用场景的性能要求之外，还考虑了成本。有关存储类定价，请参阅 Amazon S3 定价。

设置对象的存储类

要设置和更新对象存储类，您可以使用 Amazon S3 控制台、AWS SDK 或 AWS Command Line Interface（AWS CLI）。所有这些方法都使用 Amazon S3 API 操作将请求发送到 Amazon S3。

Amazon S3 API 操作支持按照以下所示方式设置（或更新）对象的存储类：

- 创建新对象时，可以指定其存储类。例如，使用 PUT Object、POST Object 和 Initiate Multipart Upload API 操作时，添加 `x-amz-storage-class` 请求标头以指定存储类。如果您未添加此标头，Amazon S3 将使用默认的 S3 Standard 存储类。
- 您还可以通过使用 PUT Object - Copy API 操作来复制对象，从而将已存储在 Amazon S3 中的对象的存储类更改为任何其他存储类。但是，您无法使用 PUT Object - Copy 来复制存储在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中的对象。您也无法从 S3 One Zone-IA 转换为 S3 Glacier 即时检索。

  通过使用相同的键名称并指定请求标头，可以复制同一个存储桶中的对象，如下所示：

  - 将 `x-amz-metadata-directive` 标头设置为 `COPY`。
  - 将 `x-amz-storage-class` 标头设置为要使用的存储类。

  在启用版本控制的存储桶中，您无法更改特定版本对象的存储类。当您复制对象时，Amazon S3 将授予它一个新的版本 ID。
- 如果对象大小小于 160 GB，则可以使用 Amazon S3 控制台更改对象的存储类。如果更大，我们建议添加 S3 生命周期配置来更改对象的存储类。
- 如果您使用 Amazon S3 控制台更改具有用户定义标签的对象的存储类，您必须拥有 `s3:GetObjectTagging` 权限。如果您要对没有用户定义标签但大小超过 16 MB 的对象更改存储类，您还必须拥有 `s3:GetObjectTagging` 权限。如果目标存储桶策略拒绝 `s3:GetObjectTagging` 操作，则将更新对象的存储类，但将从对象中移除用户定义的标签，并且您将收到错误。
- 您可以通过向存储桶添加 S3 生命周期配置来指示 Amazon S3 更改对象的存储类。有关更多信息，请参阅 管理存储生命周期。
- 在设置复制配置时，您可以将已复制对象的存储类设置为其他任何存储类。但是，您无法复制存储在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中的对象。有关更多信息，请参阅复制配置。

将访问策略权限限制到特定存储类

在授予 Amazon S3 操作的访问策略权限时，您可以使用 `s3:x-amz-storage-class` 条件键限制存储上传的对象时使用的存储类。例如，授予 `s3:PutObject` 权限时，您可以将对象上传限制到特定的存储类。有关策略示例，请参阅示例：将对象上传限制为具有特定存储类的对象。

有关在策略中使用条件的更多信息和 Amazon S3 条件键的完整列表，请参阅以下主题：

- 《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。
- 使用条件键的存储桶策略示例

使用 S3 Glacier 存储类作为长期数据存储

Amazon S3 提供多个 S3 Glacier 存储类，旨在提供经济实惠的解决方案，用来存储不经常访问的长期数据。S3 Glacier 存储类为：

- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

您可以根据访问数据的频率和需要检索数据的速度来选择其中一个存储类。这些存储类中的每一个都提供与 S3 Standard 存储类相同的持久性和弹性，但存储成本较低。有关 S3 Glacier 存储类的更多信息，请参阅 https://aws.amazon.com/s3/storage-classes/glacier/。

主题

- 比较 S3 Glacier 存储类
- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive
- 归档存储
- 这些存储类与 S3 Glacier 服务有何不同

比较 S3 Glacier 存储类

每个 S3 Glacier 存储类对于所有对象都有最短存储持续时间。如果您在最短存储持续时间之前删除、覆盖对象或将其转换为不同的存储类，则您需要按完整的最短存储持续时间付费。

有些 S3 Glacier 存储类用于归档目的，这意味着存储在这些存储类中的对象已归档，不可用于实时访问。有关更多信息，请参阅 归档存储。

专为不太频繁访问模式和较长检索时间而设计的存储类可提供较低的存储成本。有关定价信息，请参阅 https://aws.amazon.com/s3/pricing/。

下表汇总了选择 S3 Glacier 存储类时要考虑的要点：

S3 Glacier Instant Retrieval

对于每季度访问一次且需要毫秒级检索时间的长期数据，我们建议使用 S3 Glacier Instant Retrieval。这种存储类非常适合注重性能的使用案例，例如图像托管、文件共享应用程序以及存储医疗记录以供在预约期间访问。

S3 Glacier Instant Retrieval 存储类提供实时访问对象的能力，具有与 S3 Standard-IA 存储类相同的延迟和吞吐量性能。与 S3 Standard-IA 相比，S3 Glacier Instant Retrieval 的存储成本更低，但数据访问成本更高。

存储在 S3 Glacier Instant Retrieval 存储类中的数据对象大小最低为 128 KB。此存储类还具有 90 天的最短存储持续时间。

S3 Glacier Flexible Retrieval

对于每年访问一到两次且不需要立即访问的归档数据，我们建议使用 S3 Glacier Flexible Retrieval。S3 Glacier Flexible Retrieval 提供灵活的检索时间，有助于您平衡成本，访问时间从几分钟到几小时不等，并且可以免费批量检索。此存储类非常适合备份和灾难恢复。

存储在 S3 Glacier Flexible Retrieval 中的对象已归档，无法实时访问。有关更多信息，请参阅 归档存储。要访问这些对象，首先启动还原请求，该请求会创建对象的临时副本，您可以在请求完成时访问该副本。有关信息，请参阅使用归档的对象。当您还原对象时，您可以选择一个检索层来满足您的使用案例，还原时间越长，成本越低。

以下检索层适用于 S3 Glacier Flexible Retrieval：

- 加速检索 - 通常在 1-5 分钟内还原对象。加速检索视需求而定，因此，为了确保您拥有可靠且可预测的还原时间，建议您购买预置检索容量。有关更多信息，请参阅 预调配容量。
- 标准检索 - 通常在 3-5 小时内还原对象，或者当使用 S3 批量操作时，在 1 分钟到 5 小时内还原对象。有关更多信息，请参阅 使用批量操作还原对象。
- 批量检索 - 通常在 5 到 12 小时内还原对象。批量检索是免费的。

S3 Glacier Flexible Retrieval 存储类中对象的最短存储持续时间为 90 天。

S3 Glacier Flexible Retrieval 对于每个对象需要 40 KB 的额外元数据。这包括识别和检索您的数据所需的 32 KB 元数据，按 S3 Glacier Flexible Retrieval 的默认费率计费。需要额外 8 KB 数据来维护归档对象的用户定义名称和元数据，按 S3 Standard 费率计费。

S3 Glacier Deep Archive

对于每年访问不到一次的归档数据，我们建议使用 S3 Glacier Deep Archive。此存储类专为将数据集保留多年来满足合规性要求而设计，也可用于备份或灾难恢复，或任何不频繁访问的数据，您可能要等待多达 72 小时才能检索这些数据。S3 Glacier Deep Archive 是 AWS 中成本最低的存储选项。

存储在 S3 Glacier Deep Archive 中的对象已归档，无法实时访问。有关更多信息，请参阅 归档存储。要访问这些对象，首先启动还原请求，该请求会创建对象的临时副本，您可以在请求完成时访问该副本。有关信息，请参阅使用归档的对象。当您还原对象时，您可以选择一个检索层来满足您的使用案例，还原时间越长，成本越低。

以下检索层适用 S3 Glacier Deep Archive：

- 标准检索 - 通常在 12 小时内还原对象，或者当使用 S3 批量操作时，在 9–12 小时内还原对象。有关更多信息，请参阅 使用批量操作还原对象。
- 批量检索 - 通常在 48 小时内还原对象，成本仅为标准检索层的一小部分。

S3 Glacier Deep Archive 存储类中对象的最短存储持续时间为 180 天。

S3 Glacier Deep Archive 对于每个对象需要 40 KB 的额外元数据。这包括识别和检索您的数据所需的 32 KB 元数据，按 S3 Glacier Deep Archive 的默认费率计费。需要额外 8 KB 数据来维护归档对象的用户定义名称和元数据，按 S3 Standard 费率计费。

归档存储

S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 为归档存储类。这意味着，当您将对象存储在这些存储类中时，该对象将归档，无法直接访问。要访问已归档的对象，您需要为其提交还原请求，然后等待服务还原该对象。还原请求会还原对象的临时副本，并且在超过您在请求中指定的持续时间之后，删除该副本。有关更多信息，请参阅使用归档的对象。

这些存储类对于每个归档对象都需要 40 KB 的额外元数据。这包括识别和检索数据所需的 32 KB 元数据，按该存储类的默认费率计费。需要额外 8 KB 数据来维护归档对象的用户定义名称和元数据，按 S3 Standard 费率计费。

当使用分段上传来上传这些存储类中的对象时，将按照 S3 Standard 存储类费率计费。有关更多信息，请参阅 分段上传和定价。

对于每个账户中的每个 AWS 区域，在还原这些存储类中的对象时，处理对象还原请求数的速度可高达每秒 1000 个事务（TPS，Transactions Per Second）。

这些存储类与 S3 Glacier 服务有何不同

S3 Glacier 存储类是 Amazon S3 服务的一部分，将数据作为对象存储在 S3 存储桶中。您可以使用 S3 控制台，或使用 S3 API 或 SDK 以编程方式，来管理这些存储类中的对象。当您将对象存储在 S3 Glacier 存储类中时，您可以使用高级加密、对象标记和 S3 生命周期配置等 S3 功能，来协助管理数据可访问性和成本。

> **Important**
>
> 我们建议使用 Amazon S3 服务中的 S3 Glacier 存储类来存储所有长期数据。

Amazon S3 Glacier（S3 Glacier）服务是一项单独的服务，它将数据作为归档存储在保管库中。此服务不支持 Amazon S3 功能，也不为数据上传和下载操作提供控制台支持。我们建议不要使用 S3 Glacier 服务来存储您的长期数据。存储在 S3 Glacier 服务中的数据无法通过 Amazon S3 服务访问。如果您正在寻找有关 S3 Glacier 服务的信息，请参阅 Amazon S3 Glacier Developer Guide。要将数据从 Amazon S3 Glacier 服务传输到 Amazon S3 中的存储类，请参阅 AWS 解决方案库中的将数据从 Amazon S3 Glacier 保管库传输到 Amazon S3。

Amazon S3 Intelligent-Tiering

S3 Intelligent-Tiering 存储类的设计是通过在访问模式发生变化时自动将数据移动到最经济有效的访问层来优化存储成本，而不需要操作开销或对性能产生影响。每月只需支付少量的对象监控和自动化费用，S3 Intelligent-Tiering 即可监控访问模式并将未访问的对象移动到成本较低的访问层。

S3 Intelligent-Tiering 可在三个低延迟和高吞吐量访问层中自动节省存储成本。对于可以异步访问的数据，您可以选择激活 S3 Intelligent-Tiering 存储类中的自动存档功能。S3 Intelligent-Tiering 没有检索费用。如果稍后访问非频繁访问层或归档即时访问层中的对象，则自动将其移回频繁访问层。在 S3 Intelligent-Tiering 存储类中的访问层之间移动对象时，不会产生额外的分层费用。

S3 Intelligent-Tiering 是针对具有未知、不断变化或不可预测访问模式的数据推荐的存储类别，与对象大小或保留期无关，例如数据湖、数据分析和新应用程序。

有关使用 S3 Intelligent-Tiering 的信息，请参阅以下部分：

主题

S3 Intelligent-Tiering 工作原理

Amazon S3 Intelligent-Tiering 存储类会自动将对象存储在三个访问层。一个层针对频繁访问进行了优化，一个成本较低的层针对不频繁访问进行了优化，另一个低成本层针对很少访问的数据进行了优化。对于每月较低的对象监视和自动化收费，S3 Intelligent-Tiering 监控访问模式，并在连续 30 天未访问对象时自动将对象移动到非频繁访问层。在不访问 90 天后，对象将移动到存档即时访问层，而不会影响性能或运营开销。

要对于几分钟到几小时内可以访问的数据获得最低的存储成本，请激活归档功能以添加其他两个访问层。您可以将对象分层到归档访问层和/或深度归档访问层。借助归档访问，S3 Intelligent-Tiering 会将至少连续 90 天未访问的对象移动到归档访问层。借助深度归档访问，S3 Intelligent Tiering 会在至少连续 180 天无访问后将对象移动到深度归档访问层。对于这两个层，您可以根据需要配置无访问的天数。

以下操作构成了阻止将对象分层到归档访问层或深度归档访问层的访问：

- 通过 Amazon S3 控制台下载或复制对象。
- 使用 S3 批量复制调用 CopyObject、UploadPartCopy 或复制对象。在这些情况下，复制操作的源对象是分层的。
- 调用 GetObject、PutObject、RestoreObject、CompleteMultipartUpload、ListParts 或 SelectObjectContent。

例如，如果在指定的无访问天数（例如 180 天）之前通过 `SelectObjectContent` 访问了您的对象，则该操作会重置计时器。直到最后一个 `SelectObjectContent` 请求达到您指定的天数后，您的对象才会移动到归档访问层或深度归档访问层。

如果稍后访问非频繁访问层或归档即时访问层中的对象，则自动将其移回频繁访问层。

以下操作构成了自动将对象从不频繁访问层或归档即时访问层移回频繁访问层的访问：

- 通过 Amazon S3 控制台下载或复制对象。
- 使用批量复制调用 CopyObject、UploadPartCopy 或复制对象。在这些情况下，复制操作的源对象是分层的。
- 调用 GetObject、PutObject、RestoreObject、CompleteMultipartUpload 或 ListParts。

其他操作不构成自动将对象从不频繁访问层或归档即时访问层移回频繁访问层的访问：以下是此类操作的示例，而不是最终列表：

- 调用 HeadObject、GetObjectTagging、PutObjectTagging、ListObjects、ListObjectsV2 或 ListObjectVersions。
- 调用 SelectObjectContent 并不构成将对象向上分层到频繁访问层的访问。此外，它不能阻止将对象从频繁访问层向下分层到不频繁访问层，然后再分层到归档即时访问层。

通过在 PutBucketIntelligentTieringConfiguration 请求标头中指定 `INTELLIGENT-TIERING`，您可以将 S3 Intelligent-Tiering 配置为新创建数据的原定设置存储类。S3 Intelligent-Tiering 专为 99.9% 的可用性和 99.999999999% 的耐用性而设计。

> **Note**
>
> 如果对象大小小于 128 KB，则不会受监控，且不符合自动分层条件。较小的对象始终存储在频繁访问层中。

S3 Intelligent-Tiering 访问层。

下一节说明了不同的自动和可选访问层。当对象在访问层之间移动时，存储类保持不变（S3 Intelligent-Tiering）。

频繁访问层（自动）

这是任何对象创建或转换为 S3 Intelligent-Tiering 的对象开始其生命周期的默认访问层。只要对象被访问，对象就会保留在此层中。频繁访问层提供低延迟和高吞吐量性能。

不频繁访问层（自动）

如果对象在连续 30 天内仍未被访问，对象会转换到“不频繁访问层”。不频繁访问层提供低延迟和高吞吐量性能。

归档即时访问层（自动）

如果一个对象连续 90 天未被访问，该对象就会被移到“归档即时访问层”。归档即时访问层提供低延迟和高吞吐量性能。

归档访问层（可选）

S3 Intelligent-Tiering 为您提供了激活可以异步访问的数据的归档访问层的选项。激活后，归档访问层会自动存档至少连续 90 天未访问的对象。您可以将上次存档访问时间延长到最多 730 天。归档访问层的性能与 S3 Glacier Flexible Retrieval 存储类别相同。

此访问层的标准检索时间可从 3 到 5 小时不等。如果您使用 S3 批量操作启动还原请求，则还原将在几分钟内开始。有关检索选项和时间的更多信息，请参阅the section called “从 S3 Intelligent-Tiering 归档访问和深度归档访问层恢复对象”。

> **Note**
>
> 如果您想绕过归档即时访问层，则只激活 90 天的归档访问层。归档访问层提供了略低的存储成本，检索时间只需几分钟到一小时。归档即时访问层提供毫秒级的访问和高吞吐量性能。

深层归档访问层（可选）

S3 Intelligent-Tiering 为您提供了激活可以异步访问的数据的深度归档访问层的选项。激活后，深度归档访问层会自动存档至少连续 180 天未访问的对象。您可以将上次存档访问时间延长到最多 730 天。深层归档访问层的性能与 S3 Glacier Deep Archive 存储类别相同。

在 12 小时内对此访问层中的对象进行标准检索。如果您使用 S3 批量操作启动还原请求，则还原将在 9 小时内开始。有关检索选项和时间的更多信息，请参阅the section called “从 S3 Intelligent-Tiering 归档访问和深度归档访问层恢复对象”。

> **Note**
>
> 仅当您的应用程序可以异步访问对象时，才激活归档访问和深层归档访问层。如果您正在检索的对象存储在归档访问层或深度归档访问层中，必须首先使用 `RestoreObject` 操作来还原对象。

使用 S3 Intelligent-Tiering

您可以使用 S3 Intelligent-Tiering 存储类来自动优化存储成本。S3 Intelligent-Tiering 通过在访问模式更改时在访问层之间的细粒度对象级别上移动数据，实现自动成本节约。对于可以异步访问的数据，您可以使用AWS Management Console、AWS CLI 或 Amazon S3 API 选择在 S3 Intelligent-Tiering 存储类中启用自动归档。

将数据移动到 S3 Intelligent-Tiering

有两种方法可以将数据移动到 S3 Intelligent-Tiering 中。您可以通过在 `x-amz-storage-class` 标头中指定 `INTELLIGENT_TIERING` 直接将数据 PUT 到 S3 Intelligent-Tiering 中，或配置 S3 生命周期配置以将对象从 S3 Standard 或 S3 Standard-Infrequent Access 转换为 S3 Intelligent-Tiering。

使用 Direct PUT 将数据上载到 S3 Intelligent-Tiering

使用 PUT API 操作将对象上传到 S3 Intelligent-Tiering 存储类时，您可以在 `x-amz-storage-class` 请求标头中指定 S3 Intelligent-Tiering。

以下请求在 `myBucket` 桶中存储镜像 `my-image.jpg`。请求使用 `x-amz-storage-class` 标头来请求使用 S3 Intelligent-Tiering 存储类来存储对象。

Example

```
PUT /my-image.jpg HTTP/1.1
Host: myBucket.s3.<Region>.amazonaws.com (http://amazonaws.com/)
Date: Wed, 1 Sep 2021 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: INTELLIGENT_TIERING
```

使用 S3 生命周期将数据从 S3 Standard 或 S3 Standard-Infrequent Access 转换到 S3 Intelligent-Tiering

您可以在 S3 生命周期配置中添加规则以指示 Amazon S3 将对象转换为一个存储类。有关支持的转换和相关约束的信息，请参阅使用 S3 生命周期转换对象。

您可以在桶或前缀级别指定 S3 生命周期配置。在此 S3 生命周期配置规则中，筛选条件指定了一个键前缀 (`documents/`)。因此，此规则应用于带键名前缀 `documents/` 的对象，例如 `documents/doc1.txt` 和 `documents/doc2.txt`。该规则指定了 `Transition` 指示 Amazon S3 在对象创建

0 天后将其转换到 S3 Intelligent-Tiering 存储类的操作。在这种情况下，对象有资格在创建后的午夜 UTC 转换到 S3 Intelligent-Tiering。

Example

```
<LifecycleConfiguration>
  <Rule>
    <ID>ExampleRule</ID>
    <Filter>
       <Prefix>documents/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>INTELLIGENT_TIERING</StorageClass>
    </Transition>
 </Rule>
</LifecycleConfiguration>
```

启用了版本控制的存储桶会维护一个当前对象版本，以及零个或零个以上非当前对象版本。您可以为当前和非当前对象版本定义单独的生命周期规则。

有关更多信息，请参阅 生命周期配置元素。

启用 S3 Intelligent-Tiering 归档访问和深度归档访问带

为了在几分钟到几小时内就可以访问的数据上获得最低的存储开销，可以通过使用 AWS Management Console、AWS CLI 或 Amazon S3 API 创建一个桶、前缀或对象标签级配置性能来激活一个或两个归档访问层。

使用 S3 控制台

要启用 S3 Intelligent-Tiering 自动归档

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（桶）列表中，请选择您想要的桶的名称。
3. 选择属性。
4. 导航到 S3 Intelligent-Tiering 归档配置部分并选择创建配置。
5. 在归档配置设置部分中，为 S3 Intelligent-Tiering 归档配置指定描述性配置名称。

6. 在选择配置范围下，选择要使用的配置范围。您也可以使用共享前缀、对象标签或两者结合，将配置范围限制为桶内的指定对象。

   a. 要限制配置的范围，请选择使用一个或多个筛选条件限制此配置的范围。

   b. 要使用单个前缀限制配置的范围，请在前缀下输入前缀。

   c. 要使用对象标签限制配置的范围，请选择添加标签，然后为密钥输入一个值。

7. 在状态下，选择启用。
8. 在归档设置部分中，选择一个或两个要启用的归档访问层。
9. 选择创建。

使用 AWS CLI

您可以使用以下 AWS CLI 命令管理 S3 Intelligent-Tiering 生命周期配置：

- delete-bucket-intelligent-tiering-configuration
- get-bucket-intelligent-tiering-configuration
- list-bucket-intelligent-tiering-configurations
- put-bucket-intelligent-tiering-configuration

有关设置 AWS CLI 的说明，请参阅 使用 AWS CLI 进行 Amazon S3 开发。

使用 AWS CLI，您不能将配置指定为 XML 文件。您必须改为指定 JSON。以下是您可在 AWS CLI 命令中指定的示例 XML S3 Intelligent-Tiering 生命周期配置和等效 JSON。

以下示例将 S3 Intelligent-Tiering 配置为指定桶。

Example **put-bucket-intelligent-tiering-configuration**

JSON

```
{
  "Id": "string",
  "Filter": {
    "Prefix": "string",
    "Tag": {
      "Key": "string",
      "Value": "string"
    },
```

```
    "And": {
      "Prefix": "string",
      "Tags": [
        {
          "Key": "string",
          "Value": "string"
        }
        ...
      ]
    }
  },
  "Status": "Enabled"|"Disabled",
  "Tierings": [
    {
      "Days": integer,
      "AccessTier": "ARCHIVE_ACCESS"|"DEEP_ARCHIVE_ACCESS"
    }
    ...
  ]
}
```

XML

```
PUT /?intelligent-tiering&id=Id HTTP/1.1
Host: Bucket.s3.amazonaws.com
<?xml version="1.0" encoding="UTF-8"?>
<IntelligentTieringConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Id>string</Id>
   <Filter>
      <And>
         <Prefix>string</Prefix>
         <Tag>
            <Key>string</Key>
            <Value>string</Value>
         </Tag>
         ...
      </And>
      <Prefix>string</Prefix>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
   </Filter>
```

```
   <Status>string</Status>
   <Tiering>
      <AccessTier>string</AccessTier>
      <Days>integer</Days>
   </Tiering>
   ...
</IntelligentTieringConfiguration>
```

使用 PUT API 操作

您可以使用 PutBucketIntelligentTieringConfiguration 操作具体桶，每个桶最多 1000 个 S3 Intelligent-Tiering 配置。您可以使用共享前缀或对象标签，定义存储段中哪些对象符合归档访问层的条件。使用共享前缀或对象标签，您可以使各个指标筛选条件满足特定业务应用程序、工作流或内部组织的需求。您还可以灵活地激活归档访问层、深层归档访问层或两者结合。

开始使用 S3 Intelligent-Tiering

要了解有关如何使用 S3 Intelligent-Tiering 的更多信息，请参阅教程：开始使用 S3 Intelligent-Tiering。

管理 S3 Intelligent-Tiering

S3 Intelligent-Tiering 存储类可在三个低延迟和高吞吐量访问层中自动节省存储成本。它还提供可选的归档功能，对于可在几分钟到几小时内访问的数据，这些归档功能可帮助您获得云中最低的存储成本。S3 Intelligent-Tiering 存储类支持所有 Amazon S3 功能，包括以下功能：

- S3 清单，用于验证对象的访问层
- S3 复制，用于将数据复制到任何 AWS 区域
- 使用 S3 Storage Lens 查看存储使用情况和活动指标
- 服务器端加密，用于保护对象数据
- S3 对象锁定，用于防止意外删除数据
- AWS PrivateLink，用于通过虚拟私有云（VPC）中的私有端点访问 Amazon S3

标识访问层对象存储在哪个 S3 Intelligent-Tiering

要获取对象及其相应元数据的列表，包括 S3 Intelligent-Tiering 访问层，您可以使用 the section called “管理清单”。S3 清单提供旨在列出您的对象及其相应元数据的 CSV、ORC 或 Parquet 输出文件。您

可以每天或每周接收 Amazon S3 桶或共享前缀的这些清单报告。（共享前缀是指名称以通用字符串开头的对象。）

查看 S3 Intelligent-Tiering 中对象的归档状态

要在 S3 Intelligent-Tiering 存储类中的对象移到归档访问层或深度归档访问层时接收通知，您可以设置 S3 事件通知。有关更多信息，请参阅 Enabling event notifications（启用事件通知）。

Amazon S3 可以将事件通知发布到 Amazon Simple Notification Service (Amazon SNS) 主题、Amazon Simple Queue Service (Amazon SQS) 队列或 AWS Lambda 函数。有关更多信息，请参阅 Amazon S3 事件通知。

以下是 Amazon S3 发送以发布 `s3:IntelligentTiering` 事件的消息示例。有关更多信息，请参阅 the section called “事件消息结构”。

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"IntelligentTiering",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
```

```
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
            }
         },
         "intelligentTieringEventData":{
            "destinationAccessTier": "ARCHIVE_ACCESS"
            }
      }
   ]
}
```

您还可以使用 HEAD 对象请求查看对象的归档状态。如果以 S3 Intelligent-Tiering 存储类存储对象并且对象当前位于其中一个归档层中，则 HEAD 对象响应显示当前归档层。要显示归档层，请求将使用 x-amz-archive-status 标头。

以下 HEAD 对象请求返回对象（在此情况下为 *my-image.jpg*）的元数据。

Example

```
HEAD /my-image.jpg HTTP/1.1
Host: bucket.s3.region.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:02236Q3V0RonhpaBX5sCYVf1bNRuU=
```

也可以使用 HEAD 对象请求来监控 `restore-object` 请求的状态。如果正在进行归档还原，HEAD 对象响应将包括 x-amz-restore 标头。

以下是一个 HEAD 对象响应示例，显示了一个使用 S3 Intelligent-Tiering 归档且还原请求正在进行中的对象。

Example

```
HTTP/1.1 200 OK
x-amz-id-2: FSVaTMjrmBp3Izs1NnwBZeu7M19iI8UbxMbi0A8AirHANJBo+hEftBuiESACOMJp
x-amz-request-id: E5CEFCB143EB505A
Date: Fri, 13 Nov 2020 00:28:38 GMT
Last-Modified: Mon, 15 Oct 2012 21:58:07 GMT
ETag: "1accb31fcf202eba0c0f41fa2f09b4d7"
```

```
x-amz-storage-class: 'INTELLIGENT_TIERING'
x-amz-archive-status: 'ARCHIVE_ACCESS'
x-amz-restore: 'ongoing-request="true"'
x-amz-restore-request-date: 'Fri, 13 Nov 2020 00:20:00 GMT'
Accept-Ranges: bytes
Content-Type: binary/octet-stream
Content-Length: 300
Server: AmazonS3
```

从 S3 Intelligent-Tiering 归档访问和深度归档访问层恢复对象

要访问 S3 Intelligent-Tiering 存档访问层和深度存档访问层中的对象，您必须启动还原请求，然后等待对象移动到频繁访问层。有关已存档对象的更多信息，请参阅the section called “使用归档的对象”。

从归档访问层或深度归档访问层进行还原时，对象会转换回 频繁访问层中。之后，如果对象在连续 30 天仍未被访问，它会自动进入不频繁访问层。然后，在至少连续 90 天无访问后，对象会移入归档访问层。在至少连续 180 天无访问后，对象会移入深度归档访问层。有关更多信息，请参阅 the section called “S3 Intelligent-Tiering 工作原理”。

您可以使用 Amazon S3 控制台、S3 批量操作、Amazon S3 REST API、AWS SDK 或 AWS Command Line Interface（AWS CLI）还原存档的对象。有关更多信息，请参阅 the section called “使用归档的对象”。

管理存储生命周期

要管理您的对象以使其在整个生命周期内经济高效地存储，请创建 Amazon S3 生命周期配置。Amazon S3 生命周期配置是一组规则，用于定义 Amazon S3 对一组对象应用的操作。有两种类型的操作：

- Transition actions（转换操作）–这些操作将定义对象转换为另一个存储类的时间。例如，您可以选择在对象创建 30 天后将其转换为 S3 Standard-IA 存储类，或在对象创建 1 年后将其存档到 S3 Glacier 存储类。有关更多信息，请参阅 使用 Amazon S3 存储类。

  存在与生命周期转换请求关联的成本。有关定价信息，请参阅 Amazon S3 定价。
- Expiration actions（过期操作）– 这些操作将定义对象的过期时间。Amazon S3 将代表您删除过期的对象。

  仅当您使存储期限最短的存储类中的对象过期时，才会产生与生命周期过期关联的潜在成本。有关更多信息，请参阅 最小存储持续时间收费。

> **Important**
>
> 您不能使用存储桶策略来防止通过 S3 生命周期规则进行删除或转换。例如，即使您的存储桶策略拒绝所有主体的所有操作，您的 S3 生命周期配置也仍能正常发挥作用。

现有对象和新对象

当您向存储桶添加生命周期配置时，配置规则将应用到现有对象以及您在以后添加的对象。例如，如果您在今天添加带有过期操作的生命周期配置规则，而该规则使对象在创建 30 天后过期，Amazon S3 会将任何超过 30 天的现有对象加入移除队列。

账单的变化

如果在对象符合生命周期操作资格与 Amazon S3 转移或过期对象之间存在任何延迟，则在对象符合生命周期操作的资格后立即应用账单更改。例如，如果对象计划到期但 Amazon S3 没有立即使该对象到期，则不会在到期时间后向您收取存储费用。

此行为的一个例外情况是，如果您有一个生命周期规则转换为 S3 Intelligent-Tiering 存储类。在此类情况下，在对象转换为 S3 Intelligent-Tiering 存储类之前，账单不会发生更改。

有关 S3 生命周期规则的更多信息，请参阅 生命周期配置元素。

监控生命周期规则的影响

要监控由活动的生命周期规则进行的更新的效果，请参阅the section called “如何监控生命周期规则执行的操作？”。

管理对象生命周期

为明确定义了生命周期的对象定义 S3 生命周期配置规则。例如：

- 如果您将定期日志上传到一个存储桶，您的应用程序可能需要使用这些日志一个星期或一个月。之后，您可能需要删除这些日志。
- 在限定的时间段内可能需要经常访问某些文档。自此之后，这些文档很少被访问。有时，您可能不需要对这些文档进行实时访问，但是您的组织或法规可能要求您将它们存档一段特定的时间。之后，您可以删除这些文档。
- 您可以主要为了存档目的而将一些类型的数据上传到 Amazon S3。例如，您可以存档数字媒体、财务和健康记录、原始基因组序列数据、长期数据库备份，以及为遵从法规而必须保留的数据。

利用 S3 生命周期配置规则，您可以指示 Amazon S3 将对象转换为较低成本的存储类，或者存档或删除它们。

创建生命周期配置

S3 生命周期配置是 XML 文件，由一组规则组成，这些规则预定义了您希望 Amazon S3 在对象的生命周期内对对象执行的操作。

您可以使用 Amazon S3 控制台、REST API、AWS SDK 和 AWS Command Line Interface（AWS CLI）创建生命周期配置。有关更多信息，请参阅 在存储桶上设置生命周期配置。

Amazon S3 提供了一组用于在存储桶上管理生命周期配置的 REST API 操作。Amazon S3 将该配置存储为附加到存储桶的生命周期子资源。有关详细信息，请参阅：

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

有关创建生命周期配置的更多信息，请参阅以下主题：

主题

- 使用 Amazon S3 生命周期转换对象
- 即将过期的对象
- 在存储桶上设置生命周期配置
- 生命周期和其他存储桶配置
- 配置生命周期事件通知
- 生命周期配置元素
- S3 生命周期配置的示例

使用 Amazon S3 生命周期转换对象

您可以在 S3 生命周期配置中添加规则以指示 Amazon S3 将对象转换为另一个 Amazon S3 存储类。有关存储类的更多信息，请参阅 使用 Amazon S3 存储类。您何时可以以这种方式使用 S3 生命周期配置的一些示例包括以下内容：

- 当您知道对象不常访问时，您可能会将其转换为 S3 Standard-IA 存储类。

- 您可能想要将不需要实时访问的对象归档到 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类。

现有对象和新对象

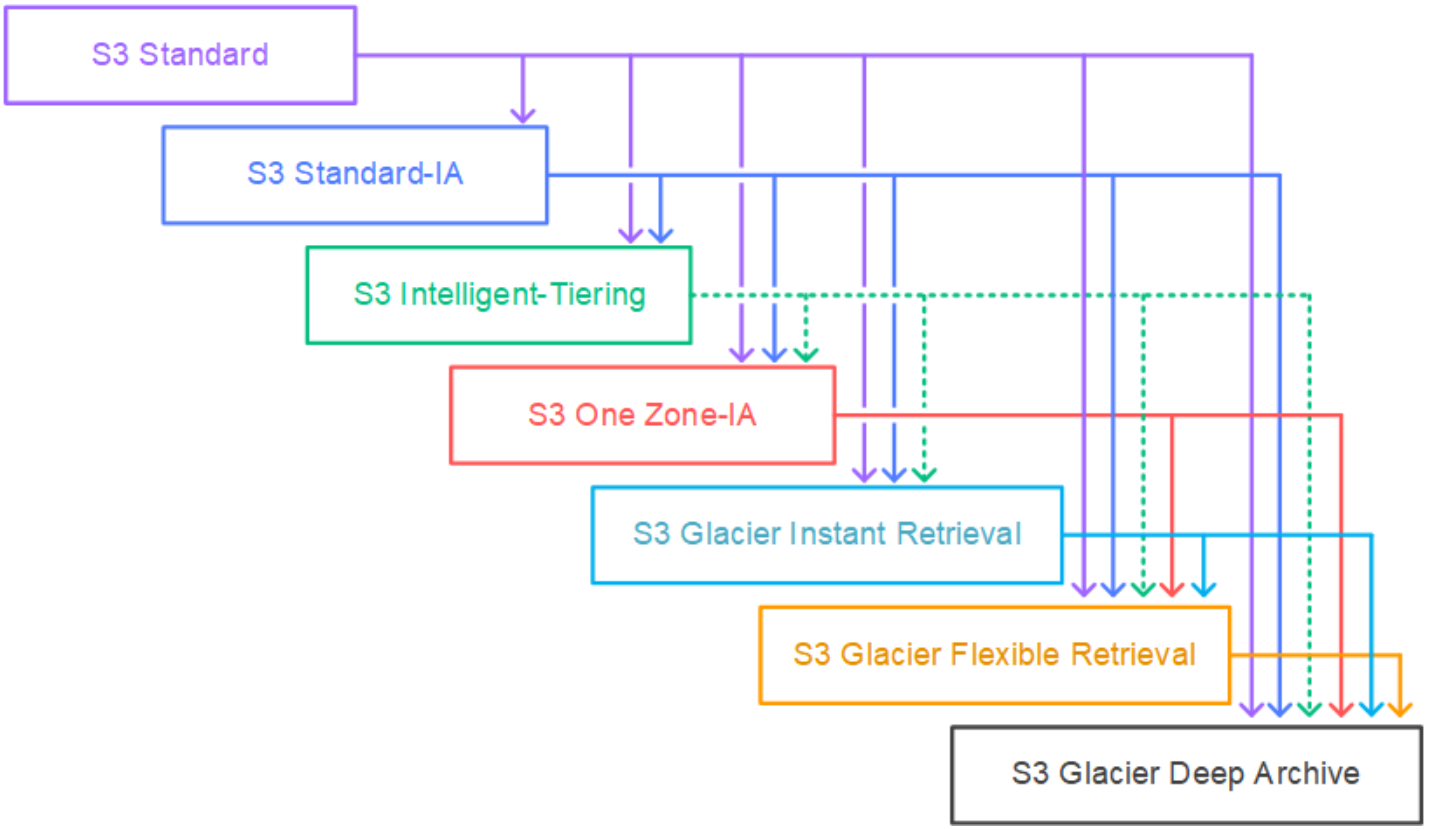
当您向存储桶添加生命周期配置时，配置规则将应用到现有对象以及您在以后添加的对象。例如，如果您在今天添加带有转换操作的生命周期配置规则，而该规则使带有特定前缀的对象在创建 30 天后转换为不同的存储类，则 Amazon S3 会将任何超过 30 天且具有指定前缀的现有对象加入转换队列。

> **Important**
>
> 您不能使用存储桶策略来防止通过 S3 生命周期规则进行删除或转换。例如，即使您的存储桶策略拒绝所有主体的所有操作，您的 S3 生命周期配置也仍能正常发挥作用。

受支持的转换和相关限制

在 S3 生命周期配置中，您可以定义用于将对象从一个存储类转换为另一个存储类的规则，以节省存储成本。如果您不了解对象的访问模式或访问模式不断变化，则可将对象转换为 S3 Intelligent-Tiering 存储类，以自动实现成本节省。有关存储类的信息，请参阅 使用 Amazon S3 存储类。

Amazon S3 支持用于在存储类之间进行转换的瀑布模型，如下图所示。



支持的生命周期转换

Amazon S3 支持在使用 S3 生命周期配置的存储类之间进行以下生命周期转换。

您可以进行以下转换：

- 从 S3 Standard 存储类转换为任何其他存储类。
- 从 S3 Standard-IA 存储类转换为 S3 Intelligent-Tiering、S3 One Zone-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类。
- 从 S3 Intelligent-Tiering 存储类转换为 S3 One Zone-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类。

> **Note**
>
> 将对象从 S3 Intelligent-Tiering 存储类转换为 S3 One Zone-IA 和一些 S3 Glacier 存储类时存在一些例外。有关更多信息，请参阅 the section called “不支持的生命周期转换”。

- 从 S3 One Zone-IA 存储类转换为 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类。

- 从 S3 Glacier Instant Retrieval 存储类转换为 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类。
- 从 S3 Glacier Flexible Retrieval 存储类转换为 S3 Glacier Deep Archive 存储类。
- 从任何存储类转换为 S3 Glacier Deep Archive 存储类。

> **Note**
>
> 生命周期转换不收取数据检索费用。但是，当使用 PUT、COPY 或生命周期规则将数据移至任何 S3 存储类时，将按请求收取摄取费用。在将对象移动到任何存储类之前，请考虑摄取或转换成本。有关成本考虑事项的更多信息，请参阅 Amazon S3 定价。

不支持的生命周期转换

Amazon S3 不支持以下任何生命周期转换。

您无法进行以下转换：

- 对于启用了版本控制或版本控制已暂停的存储桶，为任何具有 Pending 复制状态的对象。
- 从任何存储类转换为 S3 Standard 存储类。
- 任何存储类转换为低冗余存储 (RRS) 类。
- 从 S3 One Zone-IA 存储类转换为 S3 Intelligent-Tiering、S3 Standard-IA 或 S3 Glacier Instant Retrieval 存储类。
- 从 S3 Intelligent-Tiering 存储类（所有层）转换为 S3 Standard-IA 存储类。
- 从 S3 Intelligent-Tiering 存储类归档即时访问层转换为 S3 One Zone-IA。
- 从 S3 Intelligent-Tiering 存储类归档访问层转换为 S3 One Zone-IA 或 S3 Glacier Instant Retrieval。
- 从 S3 Intelligent-Tiering 存储类深层归档访问层转换为 S3 One Zone-IA、S3 Glacier Instant Retrieval 或 S3 Glacier Flexible Retrieval。

约束

生命周期存储类转换具有以下约束：

对象大小以及从 S3 Standard 或 S3 Standard-IA 到 S3 Intelligent-Tiering 、S3 Standard-IA 或 S3 One Zone-IA 的转换

当您将对象从 S3 Standard 或 S3 Standard-IA 存储类转换为 S3 Intelligent-Tiering 、S3 Standard-IA 或 S3 One Zone-IA 时，将应用以下对象大小约束：

- 较大的对象 – 对于以下转换，转换较大对象具有成本效益：
  - 从 S3 Standard 或 S3 Standard-IA 存储类转换为 S3 Intelligent-Tiering。
  - 从 S3 Standard 存储类转换为 S3 Standard-IA 或 S3 One Zone-IA。
- 小于 128KiB 的对象 – 对于以下转换，Amazon S3 不转换小于 128KiB 的对象：
  - 从 S3 Standard 或 S3 Standard-IA 存储类转换为 S3 Intelligent-Tiering 或 S3 Glacier Instant Retrieval。
  - 从 S3 Standard 存储类转换为 S3 Standard-IA 或 S3 One Zone-IA。

> **Note**
>
> 您可以根据对象大小过滤生命周期规则。

> **Important**
>
> 当您在 S3 生命周期配置中有多个规则时，对象可能变得符合条件可在同一天执行多个 S3 生命周期操作。在这种情况下，Amazon S3 遵循以下一般规则：
>
> - 永久删除优先于转换。
> - 转换优先于删除标记的创建。
> - 当对象有资格进行 S3 Glacier Flexible Retrieval 和 S3 Standard-IA（或 S3 One Zone-IA）转换时，Amazon S3 将选择 S3 Glacier Flexible Retrieval 转换。
>
> 有关示例，请参阅示例 5：重叠的筛选条件、冲突的生命周期操作，以及 Amazon S3 如何处理非版本化的存储桶。

转换为 S3 标准-IA 或 S3 单区-IA 的最短天数

在将对象转换为 S3 标准-IA 或 S3 单区-IA 之前，必须将它们存储在 Amazon S3 中至少达 30 天。例如，您无法创建用于在对象创建 1 天后将其转换为 S3 Standard-IA 存储类的生命周期规则。Amazon S3 在前 30 天内不支持此类转换，因为较新的对象的访问频率或删除速度通常高于适合 S3 标准-IA 或 S3 单区-IA 存储的值。

类似地，如果您在转换非当前对象（在受版本控制的存储桶中），则只能将至少在 30 天内是非当前版本的对象转换为 S3 Standard-IA 或 S3 One Zone-IA 存储。有关所有存储类的最短存储持续时间的列表，请参阅比较 Amazon S3 存储类。

S3 Standard-IA 和 S3 One Zone-IA 至少收取 30 天的存储费用

S3 Standard-IA 和 S3 One Zone-IA 存储等级至少收取 30 天的存储费用。因此，当 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 转换在 S3 Standard-IA 或 S3 One Zone-IA 转换发生后的不到 30 天内发生时，您无法同时为 S3 Standard-IA 或 S3 One Zone-IA 转换以及 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 转换指定单个生命周期规则。

在指定从 S3 Standard-IA 存储到 S3 One Zone-IA 的转换时，此最少 30 天存储费用也适用。您可以指定两个规则来实现这一点，而只需支付最少存储费用。有关成本考虑事项的更多信息，请参阅 Amazon S3 定价。

管理对象的完整生命周期

您可以组合这些 S3 生命周期操作来管理对象的完整生命周期。例如，假设您创建了具有明确定义的生命周期的对象。最初，这些对象在 30 天的周期内可能被经常访问。紧接着，对象在长达 90 天内不常访问。之后，不再需要对象，您可能选择存档或删除对象。

在此方案中，您创建一个 S3 生命周期规则，用于指定到 S3 Intelligent-Tiering 、S3 Standard-IA 或 S3 One Zone-IA 存储的初始转换操作，并指定到用于存档的 S3 Glacier Flexible Retrieval 存储的另一个转换操作以及一个过期操作。在将对象从一个存储类移至另一个存储类时，可节省存储成本。有关成本考虑事项的更多信息，请参阅 Amazon S3 定价。

转换为 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类（对象存档）

通过使用 S3 生命周期配置，可以将对象转换为 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类来进行存档。如果选择 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类，您的对象将在 Amazon S3 中保留。您无法直接通过单独的 Amazon S3 Glacier 服务访问它们。有关 S3 Glacier 的更多一般信息，请参阅 Amazon S3 Glacier 开发人员指南中的什么是 Amazon S3 Glacier。

在存档对象之前，请查看以下章节中的相关注意事项。

一般注意事项

下面是存档对象之前的一般注意事项：

- 加密对象在整个存储类转换过程中保持加密状态。
- 存储在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中的对象无法实时可用。

  存档的对象是 Amazon S3 对象，但在您可以访问某个存档的对象之前，您必须首先恢复它的临时副本。根据您在还原请求内指定的持续时间，还原的对象副本仅在该期间内可用。在那之后，Amazon S3 将删除临时副本，并且对象将继续归档在 S3 Glacier Flexible Retrieval 中。

  您可以使用 Amazon S3 控制台还原对象，也可以在代码中使用 AWS SDK 包装程序库或 Amazon S3 REST API 以编程方式还原对象。有关更多信息，请参阅 恢复已归档的对象。
- 存储在 S3 Glacier Flexible Retrieval 存储类中的对象只能转换为 S3 Glacier Deep Archive 存储类。

  您只能使用 S3 生命周期配置规则将对象的存储类从 S3 Glacier Flexible Retrieval 转换为 S3 Glacier Deep Archive 存储类。如果要将存储在 S3 Glacier Flexible Retrieval 中的对象的存储类更改为 S3 Glacier Deep Archive 以外的存储类，您必须首先使用还原操作制作该对象的临时副本。然后使用复制操作覆盖对象，并将 S3 Standard 、S3 Intelligent-Tiering 、S3 Standard-IA、S3 One Zone-IA 或低冗余指定为存储类。
- 对象到 S3 Glacier Deep Archive 存储类的转换只能是单向的。

  您无法使用 S3 生命周期配置规则将对象的存储类从 S3 Glacier Deep Archive 转换为任何其他存储类。如果要将已存档对象的存储类更改为其他存储类，您必须首先使用还原操作制作该对象的临时副本。然后使用复制操作覆盖对象，并将 S3 Standard、S3 Intelligent-Tiering、S3 Standard-IA、S3 One Zone-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 或 Reduced Redundancy Storage 或低冗余指定为存储类。

  > **Note**
  >
  > 在 Amazon S3 控制台中，对于 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中的对象，不支持对还原的对象执行复制操作。对于这种类型的复制操作，请使用 AWS Command Line Interface（AWS CLI）、AWS SDK 或 REST API。

  存储在 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类中的对象仅通过 Amazon S3 可见和可用。它们不能通过单独的 Amazon S3 Glacier 服务进行使用。

  这些是 Amazon S3 对象，您只能使用 Amazon S3 控制台或 Amazon S3 API 访问它们。您无法通过单独的 Amazon S3 Glacier 控制台或 Amazon S3 Glacier API 访问已存档的对象。

与成本相关的注意事项

如果您计划在数月或数年的时间内存档不经常访问的数据，则 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类可以降低您的存储成本。但是，为确保 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类适合您，请考虑以下事项：

- 存储开销费用 – 将对象转换为 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类时，需向每个对象添加固定存储量以容纳用于管理对象的元数据。
  - 对于存档到 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 的每个对象，Amazon S3 将 8 KB 存储用于对象的名称和其他元数据。Amazon S3 将存储此元数据，以便您可以使用 Amazon S3 API 获取已存档对象的实时列表。有关更多信息，请参阅获取存储桶（列出对象）。将按照 S3 Standard 标准费率对此附加存储收费。
  - 对于每个存档到 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 的对象，Amazon S3 添加 32 KB 的存储用于索引和相关元数据。标识和还原对象需要此额外数据。按照 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 费率对此附加存储收费。

  如果您打算存档小对象，请考虑这些存储费用。还可以考虑将许多小型对象合并为少量大型对象，以便减少开销成本。
- Number of days you plan to keep objects archived（计划归档对象的天数）– S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 是长期存档解决方案。S3 Glacier Flexible Retrieval 存储类的最短存储持续期间是 90 天，S3 Glacier Deep Archive 是 180 天。如果删除的对象的存档时间超过最短存储持续期间，则删除存档到 Amazon S3 Glacier 的数据是免费的。如果在最短持续期间内删除或覆盖存档的对象，则 Amazon S3 将收取按比例计算的提前删除费用。有关提前删除费用的信息，请参阅“删除存储在 Amazon S3 Glacier 中不足 90 天的对象时，如何收费？”问题（Amazon S3 常见问题）。
- S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive transition request charges（S3 Glacier 和 S3 Glacier Deep Archive 转换请求费用）– 每个转换为 S3 Glacier 或 S3 Glacier Deep Archive 存储类的对象都构成一个转换请求。每个这类请求都有成本。如果您计划转换大量对象，则考虑这些请求费用。如果您归档的对象组合包括小对象，尤其是 128KB 以下的对象，我们建议您使用生命周期对象大小筛选条件从转换中筛选出小对象，以降低请求成本。S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 不会自动阻止 128KB 以下的对象转换。
- S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive data restore charges（S3 Glacier 和 S3 Glacier Deep Archive 数据还原费用）– S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 旨在用于长期存档您不经常访问的数据。有关数据还原费用的信息，请参阅“从 Amazon S3 Glacier 检索数据如何收费？”问题（Amazon S3 常见问题）。有关如何从 Amazon S3 Glacier 还原数据的信息，请参阅恢复已归档的对象。

当通过使用 S3 生命周期管理将对象存档到 Amazon S3 Glacier 时，Amazon S3 会异步转换这些对象。S3 生命周期配置规则中的转换日期与实际转换日期之间可能存在延迟。收取的 Amazon S3 Glacier 价格基于此规则中指定的转换日期。有关更多信息，请参阅 Amazon S3 常见问题的 Amazon S3 Glacier 部分。

Amazon S3 产品详细信息页面针对存档 Amazon S3 对象提供了定价信息和示例计算。有关更多信息，请参阅以下主题：

- “将 Amazon S3 对象存档到 Amazon S3 Glacier 时，如何计算存储费用？”（Amazon S3 常见问题）。
- “删除存储在 Amazon S3 Glacier 中不足 90 天的对象时，如何收费？”（Amazon S3 常见问题）。
- “从 Amazon S3 Glacier 检索数据如何收费？”（Amazon S3 常见问题）。
- 针对不同存储类的存储成本的 Amazon S3 定价。

还原存档对象

无法实时访问已归档对象。您必须首先启动恢复请求，然后耐心等待，直到对象的临时副本根据您在请求中指定的持续时间变为可用。收到已还原对象的临时副本后，对象的存储类仍保持为 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive。（HeadObject 或 GetObject API 操作请求将返回 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 作为存储类。）

> **Note**
>
> 在还原某个归档时，您同时为归档（以 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 费率）和临时还原的副本（S3 Standard 存储费率）付费。有关定价的信息，请参阅 Amazon S3 定价。

您可以采用编程方式或使用 Amazon S3 控制台恢复对象副本。Amazon S3 针对每个对象每次仅处理一个还原请求。有关更多信息，请参阅 恢复已归档的对象。

即将过期的对象

当对象根据其生命周期配置达到其生命周期终点时，Amazon S3 会根据存储桶所处的 S3 版本控制状态采取行动。

- 不受版本控制的存储桶 – Amazon S3 会将该对象加入移除队列并异步移除该对象，从而永久移除该对象。

- 受版本控制的存储桶 – 如果当前对象版本不是删除标记，Amazon S3 将添加具有唯一的版本 ID 的删除标记。这会使当前对象版本变为非当前版本，并使删除标记变为当前版本。
- 已暂停版本控制的存储桶 – Amazon S3 创建版本 ID 为 null 的删除标记。此删除标记会在版本层次结构中将任何对象版本替换为 null 版本 ID，从而实际上删除对象。

对于受版本控制的存储桶（也即，已启用版本控制或已暂停版本控制），有一些指导 Amazon S3 如何处理过期操作的注意事项。对于启用版本控制或暂停版本控制的存储桶，以下内容适用：

- 对象过期仅适用于对象的当前版本（它对非当前对象版本没有影响）。
- 如果有一个或多个对象版本，并且删除标记是当前版本，则 Amazon S3 不会执行任何操作。
- 如果当前对象版本是唯一的对象版本并且它还是删除标记（也称为过期对象删除标记，在这种情况下，所有对象版本都已删除，您只剩下一个删除标记），则 Amazon S3 将删除过期对象删除标记。您还可以使用过期操作来指示 Amazon S3 移除所有过期对象删除标记。有关示例，请查看 示例 7：移除过期对象删除标记。
- 您可以使用 `NoncurrentVersionExpiration` 操作元素来指示 Amazon S3 何时永久删除对象的非当前版本。删除的对象无法恢复。您可以根据自对象变为非当前对象以来的一定天数来确定此到期日期。除了天数之外，还可以提供要保留的最大非当前版本数（介于 1 和 100 之间）。此值指定在 Amazon S3 可以对给定版本执行关联操作之前，必须存在多少较新的非当前版本。要指定最大非当前版本数量，您还必须提供 `Filter` 元素。如果您未指定 `Filter` 元素，Amazon S3 会在您提供最大数量的非当前版本时生成 `InvalidRequest` 错误。有关使用 `NoncurrentVersionExpiration` 操作元素的更多信息，请参阅the section called “用于描述生命周期操作的元素”。
- Amazon S3 不会对应用了 S3 对象锁定配置的非当前版本的对象采取任何操作。
- 对于具有 `Pending` 复制状态的对象，Amazon S3 不会对当前或非当前版本的对象采取任何操作。

有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

> **Important**
>
> 当您在 S3 生命周期配置中有多个规则时，对象可能变得符合条件可在同一天执行多个 S3 生命周期操作。在这种情况下，Amazon S3 遵循以下一般规则：
>
> - 永久删除优先于转换。
> - 转换优先于删除标记的创建。

- 当对象符合条件可进行 S3 Glacier Flexible Retrieval 和 S3 Standard-IA（或 S3 One Zone-IA）转换时，Amazon S3 将选择 S3 Glacier Flexible Retrieval 转换。

有关示例，请参阅示例 5：重叠的筛选条件、冲突的生命周期操作，以及 Amazon S3 如何处理非版本化的存储桶。

现有对象和新对象

当您向存储桶添加生命周期配置时，配置规则将应用到现有对象以及您在以后添加的对象。例如，如果您在今天添加带有过期操作的生命周期配置规则，而该规则使带有特定前缀的对象在创建 30 天后过期，Amazon S3 会将任何超过 30 天且具有指定前缀的现有对象加入移除队列。

Important

您不能使用存储桶策略来防止通过 S3 生命周期规则进行删除或转换。例如，即使您的存储桶策略拒绝所有主体的所有操作，您的 S3 生命周期配置也仍能正常发挥作用。

如何查找对象何时过期

要找出对象计划到期的时间，请使用 HeadObject 或 GetObject API 操作。这些 API 操作返回的响应标头提供了对象不再可供缓存的日期和时间。

Note

- 过期日期和 Amazon S3 删除对象的日期之间可能会有一段延迟。对象过期后，不会再向您收取相关的过期或存储时间费用。
- 在更新、禁用或删除生命周期规则之前，请使用 `LIST` API 操作（例如 ListObjectsV2、ListObjectVersions 和 ListMultipartUploads）或 Amazon S3 清单，根据您的应用场景验证 Amazon S3 是否已转换符合条件的对象并使其到期。

最小存储持续时间收费

如果您创建的 S3 生命周期过期规则将促使在 S3 Standard-IA 或 S3 One Zone-IA 存储中存储不到 30 天的对象过期，则您需要支付 30 天的费用。如果您创建的生命周期过期规则将促使存储在 S3 Glacier

Flexible Retrieval 存储中不到 90 天的对象过期，则您需要支付 90 天的费用。如果您创建的生命周期过期规则将促使存储在 S3 Glacier Deep Archive 存储中不到 180 天的对象过期，则您需要支付 180 天的费用。

有关更多信息，请参阅 Amazon S3 定价。

在存储桶上设置生命周期配置

本节介绍如何使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API 在存储桶上设置 Amazon S3 生命周期配置。有关 S3 生命周期配置的信息，请参阅 管理存储生命周期。

您可以使用生命周期规则来定义您希望 Amazon S3 在对象的生命周期内执行的操作（例如，将对象转化为另一个存储类、检索它们或在指定时期后删除它们）。

在设置生命周期配置之前，请注意以下情况：

生命周期配置传播延时

将 S3 生命周期配置添加到存储桶时，在将新的或更新的生命周期配置完全传播到所有 Amazon S3 系统前通常存在一些滞后。在此配置完全生效之前，预计会有几分钟延迟。在删除 S3 生命周期配置时也可能出现这种延迟。

转换或到期延迟

从满足生命周期规则到规则操作完成之间会有延迟。例如，假设一组对象根据生命周期规则在 1 月 1 日到期。尽管已在 1 月 1 日满足到期规则，但 Amazon S3 可能要等到几天甚至几周后，才会真正删除这些对象。之所以出现这种延迟，是因为 S3 生命周期以异步方式将对象排入队列来执行转换或到期操作。然而，当满足生命周期规则时，通常会应用账单的变化，即使操作尚未完成也是如此。有关更多信息，请参阅账单的变化。要监控由活动的生命周期规则进行的更新的效果，请参阅the section called “如何监控生命周期规则执行的操作？”

禁用或删除生命周期规则

禁用或删除生命周期规则时，在短暂延迟后，Amazon S3 会停止安排新对象的删除或转换。已安排的任何对象都将被取消安排，并且不会被删除或转换。

> **Note**
>
> 在更新、禁用或删除生命周期规则之前，请使用 `LIST` API 操作（例如 ListObjectsV2、ListObjectVersions 和 ListMultipartUploads）或 Amazon S3 清单，根据您的

应用场景验证 Amazon S3 是否已转换符合条件的对象并使其到期。如果您在更新、禁用或删除生命周期规则时遇到任何问题，请参阅排查 Amazon S3 生命周期问题。

现有对象和新对象

当您向存储桶添加生命周期配置时，配置规则将应用到现有对象以及您在以后添加的对象。例如，如果您在今天添加带有过期操作的生命周期配置规则，而该规则使带有特定前缀的对象在创建 30 天后过期，Amazon S3 会将任何超过 30 天且具有指定前缀的现有对象加入移除队列。

监控生命周期规则的影响

要监控由活动的生命周期规则进行的更新的效果，请参阅the section called “如何监控生命周期规则执行的操作？”

账单的变化

当满足生命周期配置规则时，可能经过一些滞后才会触发相应的操作。但是，一旦满足生命周期配置规则，账单就会立即发生变化，无论是否执行操作。

例如，在对象到期后，即使没有立即删除对象，也不会向您收取存储费用。同样，只要对象转换时间一过，就会立即按照 S3 Glacier Flexible Retrieval 存储费率向您收费，即使没有立即将该对象转换为 S3 Glacier Flexible Retrieval 存储类也是如此。

但是，生命周期转换为 S3 Intelligent-Tiering 存储类是例外。在对象转换为 S3 Intelligent-Tiering 存储类之前，账单不会发生更改。

多个或相互冲突的规则

当您在 S3 生命周期配置中有多个规则时，对象可能变得符合条件可在同一天执行多个 S3 生命周期操作。在这种情况下，Amazon S3 遵循以下一般规则：

- 永久删除优先于转换。
- 转换优先于删除标记的创建。
- 当对象符合条件可进行 S3 Glacier Flexible Retrieval 和 S3 Standard-IA（或 S3 One Zone-IA）转换时，Amazon S3 将选择 S3 Glacier Flexible Retrieval 转换。

有关示例，请参阅示例 5：重叠的筛选条件、冲突的生命周期操作，以及 Amazon S3 如何处理非版本化的存储桶。

使用 S3 控制台

您可以使用共享前缀（以通用字符串开头的对象名称）或标签为存储桶中的所有对象或一部分对象定义生命周期规则。在生命周期规则中，您可以定义特定于当前和非当前对象版本的操作。有关更多信息，请参阅下列内容：

- 管理存储生命周期
- 在 S3 存储桶中使用版本控制

创建生命周期规则

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要为其创建生命周期规则的存储桶的名称。
3. 请选择 Management（管理）选项卡，然后选择 Create lifecycle rule（创建生命周期规则）。
4. 在 Lifecycle rule name (生命周期规则名称) 中，输入规则的名称。

   在该存储桶内，此名称必须是唯一的。
5. 请选择生命周期规则的范围：
   - 要将此生命周期规则应用于所有带特定前缀或标签的对象，请选择将范围限制在特定前缀或标签。
     - 要按前缀限制范围，请在 Prefix (前缀) 中输入前缀。
     - 要按标签限制范围，请选择 Add tag (添加标签)，然后输入标签键和值。

     有关对象名称前缀的更多信息，请参阅 创建对象键名称。有关对象标签的更多信息，请参阅 使用标签对存储进行分类。
   - 要将此生命周期规则应用于存储桶中的所有对象，请选择此规则适用于存储桶中的所有 对象，然后选择我确认此规则适用于存储桶中的所有对象。
6. 要按对象大小筛选规则，可以选择指定最小对象大小和/或指定最大对象大小选项。
   - 当您为最小对象大小或最大对象大小指定值时，该值必须大于 0 字节且不超过 5 TB。您可以以字节、KB、MB 或 GB 为单位指定此值。
   - 当您指定这两个值时，最大对象大小必须大于最小对象大小。

> **Note**
>
> 最小对象大小和最大对象大小筛选条件不包含所指定的值。例如，如果您将筛选条件设置为使最小对象大小为 128 KB 的对象到期，则恰好为 128 KB 的对象不会到期。相反，该规则仅适用于大小超过 128 KB 的对象。

7. 在 Lifecycle rule actions (生命周期规则操作) 下，请选择希望生命周期规则执行的操作：

   - 在存储类之间转换对象的当前版本
   - 在存储类之间转换对象的先前版本
   - 使对象的当前版本过期

   > **Note**
   >
   > 对于未启用 S3 版本控制的存储桶，使当前版本到期会导致 Amazon S3 永久删除对象。有关更多信息，请参阅 the section called “生命周期操作与桶版本控制状态”。

   - 永久删除对象的先前版本
   - 删除过期的删除标记或未完成的分段上传

   根据您选择的操作，会显示不同的选项。

8. 要在存储类之间转换对象的当前版本，请在 Transition current versions of objects between storage classes (在存储类之间转换对象的当前版本) 下面：

   a. 在存储类转换中，请选择要转换到的存储类。有关可能的转换的列表，请参阅the section called “支持的生命周期转换”。可从以下存储类中进行选择：

      - S3 标准 - IA
      - S3 智能分层
      - S3 单区 - IA
      - S3 Glacier Flexible Retrieval
      - S3 Glacier Deep Archive

   b. 在 Days after object creation (对象创建后的天数) 中，输入创建后转换对象的天数。

有关存储类的更多信息，请参阅 使用 Amazon S3 存储类。您可以为当前对象版本和/或之前的对象版本定义转换。版本控制允许您在一个存储桶中保留多个版本的对象。有关版本控制的更多信息，请参阅 使用 S3 控制台。

> **Important**
>
> 如果选择 S3 Glacier Flexible Retrieval 或 Glacier Deep Archive 存储类，您的对象将在 Amazon S3 中保留。您无法直接通过单独的 Amazon S3 Glacier 服务访问它们。有关更多信息，请参阅 使用 Amazon S3 生命周期转换对象。

9. 要在存储类之间转换对象的非当前 版本，请在在存储类之间转换对象的非当前版本下面：

   a. 在存储类转换中，请选择要转换到的存储类。有关可能的转换的列表，请参阅the section called “支持的生命周期转换”。可从以下存储类中进行选择：

      - S3 标准 - IA
      - S3 智能分层
      - S3 单区 - IA
      - S3 Glacier Flexible Retrieval
      - S3 Glacier Deep Archive

   b. 在对象成为非当前对象后的天数中，输入创建后转换对象的天数。

10. 要使对象的当前版本过期，请在 Expire current versions of objects（使对象的当前版本过期）下面的 Number of days after object creation（对象创建后的天数）中输入天数。

> **Important**
>
> 在不受版本控制的存储桶中，到期操作会导致 Amazon S3 永久移除该对象。有关生命周期操作的更多信息，请参阅 用于描述生命周期操作的元素。

11. 要永久删除对象的先前版本，请在 Permanently delete noncurrent versions of objects（永久删除对象的先前版本）下的 Days after objects become noncurrent（对象变为非当前对象后的天数）中输入天数。您可以选择在 Number of newer versions to retain（要保留的较新版本的数量）下输入一个值，从而指定要保留的较新版本数量。

12. 在 Delete expired delete markers or incomplete multipart uploads (删除过期的删除标记或未完成的分段上传) 下面，请选择 Delete expired object delete markers (删除过期对象的删除标记) 和

Delete incomplete multipart uploads (删除未完成的分段上传)。然后，输入您要在分段上传启动多少天后结束并清理未完成的分段上传。

有关分段上传的更多信息，请参阅 使用分段上传来上传和复制对象。

13. 请选择 Create rule (创建规则)。

如果规则没有任何错误，Amazon S3 会启用它，并且您可以在 Lifecycle rules (生命周期规则) 下的 Management (管理) 选项卡上看到它。

有关 AWS CloudFormation 模板和示例的信息，请参阅《AWS CloudFormation 用户指南》中的使用 AWS CloudFormation 模板和 AWS::S3::Bucket。

使用 AWS CLI

您可以使用以下 AWS CLI 命令管理 S3 生命周期配置：

- `put-bucket-lifecycle-configuration`
- `get-bucket-lifecycle-configuration`
- `delete-bucket-lifecycle`

有关设置 AWS CLI 的说明，请参阅 使用 AWS CLI 进行 Amazon S3 开发。

Amazon S3 生命周期配置是一个 XML 文件。但在使用 AWS CLI 时，您不能指定 XML 格式。您必须改为指定 JSON 格式。以下是您可在 AWS CLI 命令中指定的示例 XML 生命周期配置和等效 JSON 配置。

请考虑以下示例 S3 生命周期配置。

Example 示例 1

Example

XML

```
<LifecycleConfiguration>
    <Rule>
        <ID>ExampleRule</ID>
        <Filter>
           <Prefix>documents/</Prefix>
        </Filter>
        <Status>Enabled</Status>
```

```
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "Filter": {
                "Prefix": "documents/"
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 365,
                    "StorageClass": "GLACIER"
                }
            ],
            "Expiration": {
                "Days": 3650
            },
            "ID": "ExampleRule"
        }
    ]
        }
```

Example 示例 2

Example

XML

```
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Rule>
```

```
        <ID>id-1</ID>
        <Expiration>
            <Days>1</Days>
        </Expiration>
        <Filter>
            <And>
                <Prefix>myprefix</Prefix>
                <Tag>
                    <Key>mytagkey1</Key>
                    <Value>mytagvalue1</Value>
                </Tag>
                <Tag>
                    <Key>mytagkey2</Key>
                    <Value>mytagvalue2</Value>
                </Tag>
            </And>
        </Filter>
        <Status>Enabled</Status>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ]
                }
            },
```

```
            "Status": "Enabled",
            "Expiration": {
                "Days": 1
            }
        }
    ]
}
```

您可以测试 put-bucket-lifecycle-configuration，如下所示。

测试配置

1. 将 JSON 生命周期配置保存在一个文件（例如 *lifecycle.json*）中。
2. 运行以下 AWS CLI 命令以在存储桶上设置生命周期配置。将 *user input placeholders* 替换为您自己的信息。

```
$ aws s3api put-bucket-lifecycle-configuration  \
--bucket amzn-s3-demo-bucket  \
--lifecycle-configuration file://lifecycle.json
```

3. 要进行验证，请使用 get-bucket-lifecycle-configuration AWS CLI 命令检索 S3 生命周期配置，如下所示：

```
$ aws s3api get-bucket-lifecycle-configuration  \
--bucket amzn-s3-demo-bucket
```

4. 要删除 S3 生命周期配置，请使用 delete-bucket-lifecycle AWS CLI 命令，如下所示：

```
aws s3api delete-bucket-lifecycle \
--bucket amzn-s3-demo-bucket
```

使用 AWS SDK

Java

您可以使用 AWS SDK for Java 管理存储桶的 S3 生命周期配置。有关管理 S3 生命周期配置的更多信息，请参阅 管理存储生命周期。

> **Note**
>
> 在将 S3 生命周期配置添加到存储桶时，Amazon S3 会替换存储桶的当前生命周期配置（如果有）。要更新一个配置，请检索它，进行所需的更改，然后向存储桶添加已修订的配置。

以下示例说明如何使用 AWS SDK for Java 添加、更新和删除存储桶的生命周期配置。本示例执行以下操作：

- 向存储桶添加生命周期配置。
- 检索生命周期配置并通过添加其他规则来更新该配置。
- 向存储桶添加已修改的生命周期配置。Amazon S3 将替换现有配置。
- 再次检索配置，并通过输出规则数来验证它是否具有正确数量的规则。
- 删除生命周期配置并通过再次尝试检索它来验证它是否已被删除。

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration;
import com.amazonaws.services.s3.model.BucketLifecycleConfiguration.Transition;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.Tag;
import com.amazonaws.services.s3.model.lifecycle.LifecycleAndOperator;
import com.amazonaws.services.s3.model.lifecycle.LifecycleFilter;
import com.amazonaws.services.s3.model.lifecycle.LifecyclePrefixPredicate;
import com.amazonaws.services.s3.model.lifecycle.LifecycleTagPredicate;

import java.io.IOException;
import java.util.Arrays;

public class LifecycleConfiguration {

        public static void main(String[] args) throws IOException {
```

```
                Regions clientRegion = Regions.DEFAULT_REGION;
                String bucketName = "*** Bucket name ***";

                // Create a rule to archive objects with the "glacierobjects/"
prefix to Glacier
                // immediately.
                BucketLifecycleConfiguration.Rule rule1 = new
BucketLifecycleConfiguration.Rule()
                                .withId("Archive immediately rule")
                                .withFilter(new LifecycleFilter(new
LifecyclePrefixPredicate("glacierobjects/")))
                                .addTransition(new
Transition().withDays(0).withStorageClass(StorageClass.Glacier))
                                .withStatus(BucketLifecycleConfiguration.ENABLED);

                // Create a rule to transition objects to the Standard-Infrequent
Access storage
                // class
                // after 30 days, then to Glacier after 365 days. Amazon S3 will
delete the
                // objects after 3650 days.
                // The rule applies to all objects with the tag "archive" set to
"true".
                BucketLifecycleConfiguration.Rule rule2 = new
BucketLifecycleConfiguration.Rule()
                                .withId("Archive and then delete rule")
                                .withFilter(new LifecycleFilter(new
LifecycleTagPredicate(new Tag("archive", "true"))))
                                .addTransition(new Transition().withDays(30)

.withStorageClass(StorageClass.StandardInfrequentAccess))
                                .addTransition(new
Transition().withDays(365).withStorageClass(StorageClass.Glacier))
                                .withExpirationInDays(3650)
                                .withStatus(BucketLifecycleConfiguration.ENABLED);

                // Add the rules to a new BucketLifecycleConfiguration.
                BucketLifecycleConfiguration configuration = new
BucketLifecycleConfiguration()
                                .withRules(Arrays.asList(rule1, rule2));

                try {
                        AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
```

```
                                        .withCredentials(new
ProfileCredentialsProvider())
                                        .withRegion(clientRegion)
                                        .build();

                        // Save the configuration.
                        s3Client.setBucketLifecycleConfiguration(bucketName,
configuration);

                        // Retrieve the configuration.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);

                        // Add a new rule with both a prefix predicate and a tag
predicate.
                        configuration.getRules().add(new
BucketLifecycleConfiguration.Rule().withId("NewRule")
                                        .withFilter(new LifecycleFilter(new
LifecycleAndOperator(
                                                        Arrays.asList(new
LifecyclePrefixPredicate("YearlyDocuments/"),
                                                                        new
LifecycleTagPredicate(new Tag(

   "expire_after",

   "ten_years"))))))
                                        .withExpirationInDays(3650)

.withStatus(BucketLifecycleConfiguration.ENABLED));

                        // Save the configuration.
                        s3Client.setBucketLifecycleConfiguration(bucketName,
configuration);

                        // Retrieve the configuration.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);

                        // Verify that the configuration now has three rules.
                        configuration =
s3Client.getBucketLifecycleConfiguration(bucketName);
                        System.out.println("Expected # of rules = 3; found: " +
configuration.getRules().size());
```

```
                        // Delete the configuration.
                        s3Client.deleteBucketLifecycleConfiguration(bucketName);

                        // Verify that the configuration has been deleted by
 attempting to retrieve it.
                        configuration =
 s3Client.getBucketLifecycleConfiguration(bucketName);
                        String s = (configuration == null) ? "No configuration
 found." : "Configuration found.";
                        System.out.println(s);
                } catch (AmazonServiceException e) {
                        // The call was transmitted successfully, but Amazon S3
 couldn't process
                        // it, so it returned an error response.
                        e.printStackTrace();
                } catch (SdkClientException e) {
                        // Amazon S3 couldn't be contacted for a response, or the
 client
                        // couldn't parse the response from Amazon S3.
                        e.printStackTrace();
                }
        }
}
```

.NET

您可以使用 AWS SDK for .NET 管理存储桶上的 S3 生命周期配置。有关管理生命周期配置的更多信息，请参阅 管理存储生命周期。

> **Note**
>
> 在添加生命周期配置时，Amazon S3 将替换指定存储桶上的现有配置。要更新配置，您必须先检索生命周期配置，进行更改，然后向存储桶添加已修订的生命周期配置。

以下示例说明如何使用 AWS SDK for .NET 添加、更新和删除存储桶的生命周期配置。该代码示例执行以下操作：

- 向存储桶添加生命周期配置。
- 检索生命周期配置并通过添加其他规则来更新该配置。

- 向存储桶添加已修改的生命周期配置。Amazon S3 将替换现有生命周期配置。
- 再次检索配置并通过输出配置中的规则数进行验证。
- 删除生命周期配置并确认删除操作。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class LifecycleTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddUpdateDeleteLifecycleConfigAsync().Wait();
        }

        private static async Task AddUpdateDeleteLifecycleConfigAsync()
        {
            try
            {
                var lifeCycleConfiguration = new LifecycleConfiguration()
                {
                    Rules = new List<LifecycleRule>
                        {
                            new LifecycleRule
                            {
                                 Id = "Archive immediately rule",
                                 Filter = new LifecycleFilter()
                                 {
```

```
                                    LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                    {
                                        Prefix = "glacierobjects/"
                                    }
                                },
                                Status = LifecycleRuleStatus.Enabled,
                                Transitions = new List<LifecycleTransition>
                                {
                                     new LifecycleTransition
                                     {
                                          Days = 0,
                                          StorageClass = S3StorageClass.Glacier
                                     }
                                 },
                            },
                            new LifecycleRule
                            {
                                 Id = "Archive and then delete rule",
                                  Filter = new LifecycleFilter()
                                 {
                                    LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                    {
                                        Prefix = "projectdocs/"
                                    }
                                 },
                                 Status = LifecycleRuleStatus.Enabled,
                                 Transitions = new List<LifecycleTransition>
                                 {
                                      new LifecycleTransition
                                      {
                                           Days = 30,
                                           StorageClass =
S3StorageClass.StandardInfrequentAccess
                                      },
                                      new LifecycleTransition
                                      {
                                        Days = 365,
                                        StorageClass = S3StorageClass.Glacier
                                      }
                                 },
                                 Expiration = new LifecycleRuleExpiration()
                                 {
```

```
                                    Days = 3650
                                }
                            }
                        }
                };

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Retrieve an existing configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

                // Add a new rule.
                lifeCycleConfiguration.Rules.Add(new LifecycleRule
                {
                    Id = "NewRule",
                    Filter = new LifecycleFilter()
                    {
                        LifecycleFilterPredicate = new LifecyclePrefixPredicate()
                        {
                            Prefix = "YearlyDocuments/"
                        }
                    },
                    Expiration = new LifecycleRuleExpiration()
                    {
                        Days = 3650
                    }
                });

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Verify that there are now three rules.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
                Console.WriteLine("Expected # of rulest=3; found:{0}",
lifeCycleConfiguration.Rules.Count);

                // Delete the configuration.
                await RemoveLifecycleConfigAsync(client);

                // Retrieve a nonexistent configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
```

```
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
LifecycleConfiguration configuration)
        {

            PutLifecycleConfigurationRequest request = new
PutLifecycleConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }

        static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client)
        {
            GetLifecycleConfigurationRequest request = new
GetLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }

        static async Task RemoveLifecycleConfigAsync(IAmazonS3 client)
        {
            DeleteLifecycleConfigurationRequest request = new
DeleteLifecycleConfigurationRequest
```

```
            {
                BucketName = bucketName
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
    }
}
```

Ruby

通过使用类 AWS::S3::BucketLifecycleConfiguration，您可使用 AWS SDK for Ruby 管理存储桶上的 S3 生命周期配置。有关管理生命周期配置的更多信息，请参阅 管理存储生命周期。

使用 REST API

《Amazon Simple Storage Service API 参考》中的以下几节描述了与 S3 生命周期配置相关的 REST API。

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

S3 生命周期问题排查

有关使用 S3 生命周期时可能出现的常见问题，请参阅the section called “排查生命周期问题”。

生命周期和其他存储桶配置

除了 S3 生命周期配置之外，您还可以将其他配置与存储桶关联。本部分解释了 S3 生命周期配置如何与其他存储桶配置相关。

生命周期和版本控制

您可以向不受版本控制的存储桶和启用了版本控制的存储桶添加 S3 生命周期配置。有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

启用了版本控制的存储桶会维护一个当前对象版本，以及零个或零个以上非当前对象版本。您可以为当前和非当前对象版本定义单独的生命周期规则。

有关更多信息，请参阅 生命周期配置元素。

> **Important**
>
> 当您在 S3 生命周期配置中有多个规则时，对象可能变得符合条件可在同一天执行多个 S3 生命周期操作。在这种情况下，Amazon S3 遵循以下一般规则：
>
> - 永久删除优先于转换。
> - 转换优先于删除标记的创建。
> - 当对象有资格进行 S3 Glacier Flexible Retrieval 和 S3 Standard-IA（或 S3 One Zone-IA）转换时，Amazon S3 将选择 S3 Glacier Flexible Retrieval 转换。
>
> 有关示例，请参阅示例 5：重叠的筛选条件、冲突的生命周期操作，以及 Amazon S3 如何处理非版本化的存储桶。

启用了 MFA 的存储桶上的生命周期配置

启用了多重身份验证（MFA）的存储桶上不支持生命周期配置。

生命周期和日志记录

AWS CloudTrail 对象级别日志记录不会捕获 Amazon S3 生命周期操作。CloudTrail 捕获向外部 Amazon S3 端点发出的 API 请求，而 S3 生命周期操作则使用内部 Amazon S3 端点执行。可以在 S3 存储桶中启用 Amazon S3 服务器访问日志，以捕获与 S3 生命周期相关的操作，例如对象转换为另一个存储类以及导致永久删除或逻辑删除的对象过期。有关更多信息，请参阅 the section called “记录服务器访问”。

如果您在存储桶上启用了日志记录，Amazon S3 服务器访问日志会报告以下操作的结果。

操作日志	描述
`S3.EXPIRE.OBJECT`	Amazon S3 由于生命周期到期操作而永久地删除对象。
`S3.CREATE.DELETEMARKER`	Amazon S3 在逻辑上删除现存版本，并在启用版本控制的存储桶中添加一个删除标记。

操作日志	描述
S3.TRANSITION_SIA.OBJECT	Amazon S3 将对象转换为 S3 Standard-IA 存储类。
S3.TRANSITION_ZIA.OBJECT	Amazon S3 将对象转换为 S3 One Zone-IA 存储类。
S3.TRANSITION_INT.OBJECT	Amazon S3 将对象转换为 S3 Intelligent-Tiering 存储类。
S3.TRANSITION_GIR.OBJECT	Amazon S3 启动将对象转换为 S3 Glacier Instant Retrieval 存储类的过程。
S3.TRANSITION.OBJECT	Amazon S3 启动将对象转换为 S3 Glacier Flexible Retrieval 存储类的过程。
S3.TRANSITION_GDA.OBJECT	Amazon S3 启动将对象转换为 S3 Glacier Deep Archive 存储类的过程。
S3.DELETE.UPLOAD	Amazon S3 中止未完成的分段上传。

> **Note**
>
> Amazon S3 服务器访问日志记录一般会尽最大努力记录信息，但不能用于完整记录所有 Amazon S3 请求。

S3 生命周期问题排查

有关排查常见的 S3 生命周期问题的更多信息，请参阅排查 Amazon S3 生命周期问题。

更多信息

- 生命周期配置元素
- 转换为 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类（对象存档）
- 在存储桶上设置生命周期配置

配置生命周期事件通知

您可以设置 Amazon S3 事件通知，以便在 Amazon S3 删除对象或按照 S3 生命周期规则将其转换为另一个 Amazon S3 存储类时接收通知。

通过使用 `LifecycleExpiration` 事件类型，每当 Amazon S3 基于 S3 生命周期配置删除对象时，您都可以收到通知。`s3:LifecycleExpiration:Delete` 事件类型在删除未转换存储桶中的对象时通知您。当 S3 生命周期配置永久删除对象版本时，它也会通知您。在删除受版本控制的存储桶中对象的当前版本后，如果 S3 生命周期创建删除标记，`s3:LifecycleExpiration:DeleteMarkerCreated` 事件类型会通知您。有关更多信息，请参阅 Delete object version（删除对象版本）。

通过使用 `s3:LifecycleTransition` 事件类型，当对象通过 S3 生命周期配置从一个 Amazon S3 存储类转换到另一个存储类时，您可以收到通知。

Amazon S3 可以将事件通知发布到 Amazon Simple Notification Service (Amazon SNS) 主题、Amazon Simple Queue Service (Amazon SQS) 队列或 AWS Lambda 函数。有关更多信息，请参阅 Amazon S3 事件通知。

有关如何配置 Amazon S3 事件通知的说明，请参阅 Enabling event notifications（启用事件通知）。

以下是 Amazon S3 发送以发布 `s3:LifecycleExpiration:Delete` 事件的消息示例。有关详细信息，请参阅 Event message structure（事件消息结构）。

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"LifecycleExpiration:Delete",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
```

```
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"amzn-s3-demo-bucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::amzn-s3-demo-bucket"
            },
            "object":{
               "key":"expiration/delete",
               "sequencer":"0055AED6DCD90281E5",
            }
         }
      }
   ]
}
```

Amazon S3 为发布 `s3:LifecycleTransition` 事件而发送的消息中还包括以下信息。

```
"lifecycleEventData":{
    "transitionEventData": {
        "destinationStorageClass": the destination storage class for the object
    }
}
```

生命周期配置元素

主题

- ID 元素
- Status 元素
- Filter 元素
- 用于描述生命周期操作的元素

您可以将 Amazon S3 生命周期配置指定为 XML，该配置包含一个或多个生命周期规则。

```
<LifecycleConfiguration>
    <Rule>
```

```
        ...
    </Rule>
    <Rule>
        ...
    </Rule>
</LifecycleConfiguration>
```

每个规则由以下内容组成：

- 规则元数据，包含规则 ID 以及指示规则是已启用还是已禁用的状态。如果规则处于禁用状态，则 Amazon S3 不会执行规则中指定的任何操作。
- 标识规则应用到的对象的筛选条件。您可以通过使用对象大小、对象键前缀、一个或多个对象标签或者筛选条件的组合来指定筛选条件。
- 您希望 Amazon S3 执行指定操作时的一个或多个转换或过期操作，带有位于对象的生命周期内的日期或时间段。

以下部分介绍了 S3 生命周期配置中的 XML 元素。有关示例配置，请参阅 S3 生命周期配置的示例。

ID 元素

一个 S3 生命周期配置最多可以有 1000 个规则。此限制不可调整。`<ID>` 元素唯一地标识规则。ID 长度最多为 255 个字符。

Status 元素

`<Status>` 元素值可以是 `Enabled` 或 `Disabled`。如果规则处于禁用状态，则 Amazon S3 不会执行规则中定义的任何操作。

Filter 元素

生命周期规则可基于您在该规则中指定的 `<Filter>` 元素，应用于存储桶中的所有对象或一部分对象。

您可以按键前缀、对象标签或二者的组合（在此情况下，Amazon S3 使用逻辑 AND 组合筛选条件）筛选对象。考虑以下示例：

- 使用键前缀指定筛选条件 – 此示例显示一个 S3 生命周期规则，此规则基于键名前缀（`logs/`）应用于一部分对象。例如，此生命周期规则应用于对象 `logs/mylog.txt`、`logs/temp1.txt` 和 `logs/test.txt`。此规则不应用于对象 `example.jpg`。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>logs/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
    ...
</LifecycleConfiguration>
```

如果您要基于不同的键名称前缀将生命周期操作应用于一部分对象，可指定单独的规则。在每个规则中，指定基于前缀的筛选条件。例如，要描述具有键前缀 projectA/ 和 projectB/ 的对象的生命周期操作，可指定两个规则，如下所示：

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>projectA/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>

    <Rule>
        <Filter>
           <Prefix>projectB/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
</LifecycleConfiguration>
```

有关对象键的更多信息，请参阅 创建对象键名称。

- 指定基于对象标签的筛选条件 – 在以下示例中，此生命周期规则指定基于标签（*key*）和值（*value*）的筛选条件。随后，此规则仅应用于具有特定标签的一部分对象。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
```

```
            <Tag>
               <Key>key</Key>
               <Value>value</Value>
            </Tag>
         </Filter>
         transition/expiration actions
         ...
    </Rule>
</LifecycleConfiguration>
```

您可以指定基于多个标签的筛选条件。您必须在 <And> 元素中包含标签，如以下示例所示。此规则指示 Amazon S3 对具有两个标签（带特定的标签键和值）的对象执行生命周期操作。

```
<LifecycleConfiguration>
    <Rule>
      <Filter>
         <And>
            <Tag>
               <Key>key1</Key>
               <Value>value1</Value>
            </Tag>
            <Tag>
               <Key>key2</Key>
               <Value>value2</Value>
            </Tag>
             ...
          </And>
      </Filter>
      transition/expiration actions
    </Rule>
</Lifecycle>
```

此生命周期规则应用于具有指定的两个标签的对象。Amazon S3 执行逻辑 AND。请注意以下几点：

- 每个标签的键和值都 必须完全匹配。如果您只指定 <Key> 元素而不指定 <Value> 元素，则该规则将仅应用于与标签键匹配而未 指定值的对象。
- 此规则应用于具有规则中指定的所有标签的一部分对象。如果某个对象指定了其他标签，则该规则仍将适用。

> **Note**
>
> 当您在筛选条件中指定多个标签时，每个标签键必须是唯一的。

- 指定基于前缀和一个或多个标签的筛选条件 – 在生命周期规则中，您可以指定基于键前缀和一个或多个标签的筛选条件。同样，您必须在 <And> 元素中包含所有这些筛选条件元素，如下所示：

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
          <And>
             <Prefix>key-prefix</Prefix>
             <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
             </Tag>
             <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
             </Tag>
              ...
          </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

  Amazon S3 使用逻辑 AND 组合这些筛选条件。也就是说，该规则应用于具有指定键前缀和指定标签的对象子集。一个筛选条件只能有一个前缀以及零个或多个标签。

- 您可以指定 empty filter（空筛选条件），在此情况下，此规则应用于桶中的所有对象。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

- 要按照 object size（对象大小）筛选规则，您可以指定最小的大小 (ObjectSizeGreaterThan) 或最大大小 (ObjectSizeLessThan)，也可以指定一系列对象大小。

  对象大小值以字节为单位。最大筛选条件大小为 5TB。某些存储类具有最小对象大小限制。有关更多信息，请参阅 比较 Amazon S3 存储类。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> ObjectSizeGreaterThan 和 ObjectSizeLessThan 筛选条件不包含所指定的值。例如，如果您将大小为 128 KB 到 1024 KB 的对象设置为从 S3 Standard 存储类移到 S3 Standard-IA 存储类，则大小精确为 1024 KB 和 128 KB 的对象将不会转换到 S3 Standard-IA。相反，该规则将仅适用于大小大于 128 KB 且小于 1024 KB 的对象。

如果要指定对象大小范围，ObjectSizeGreaterThan 整数必须小于 ObjectSizeLessThan 值。当使用多个筛选条件时，必须将筛选条件包装在 <And> 元素中。以下示例显示了如何指定范围在 500 字节到 64000 字节之间的对象。

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <Prefix>key-prefix</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
```

```
</LifecycleConfiguration>
```

用于描述生命周期操作的元素

通过在 S3 生命周期规则中指定一个或多个以下预定义操作，您可以指示 Amazon S3 在对象的生命周期内执行特定操作。这些操作的效果取决于桶的版本控制状态。

- **`Transition`** 操作元素 – 您指定 `Transition` 操作来将对象从一个存储类转换为另一个存储类。有关转换对象的更多信息，请参阅 受支持的转换和相关限制。达到对象生命周期内指定的日期或时间段后，Amazon S3 将执行该转换。

  对于受版本控制的桶（已启用版本控制或已暂停版本控制的桶），`Transition` 操作适用于当前对象版本。为了管理非当前版本，Amazon S3 定义了 `NoncurrentVersionTransition` 操作（本主题稍后会对此描述）。

- **`Expiration`** 操作元素 – `Expiration` 操作使规则中标识的对象到期，并且应用于任意 Amazon S3 存储类中符合条件的对象。有关存储类别的更多信息，请参阅 使用 Amazon S3 存储类。Amazon S3 使所有过期的对象不可用。是否永久删除对象取决于桶的版本控制状态。
  - 不受版本控制的存储桶 – `Expiration` 操作导致 Amazon S3 永久移除对象。
  - 受版本控制的桶 – 对于受版本控制的桶（即，已启用版本控制或已暂停版本控制），有一些指导 Amazon S3 如何处理 `Expiration` 操作的注意事项。对于启用版本控制或暂停版本控制的桶，以下内容适用：
    - `Expiration` 操作仅适用于当前版本 (对非当前对象版本没有影响)。
    - 如果有一个或多个对象版本，并且删除标记是当前版本，则 Amazon S3 不会执行任何操作。
    - 如果当前对象版本是唯一的对象版本并且它还是删除标记（也称为过期对象删除标记，在这种情况下，所有对象版本都已删除，您只剩下一个删除标记），则 Amazon S3 将删除过期对象删除标记。您还可以使用过期操作来指示 Amazon S3 移除所有过期对象删除标记。有关示例，请参阅 示例 7：移除过期对象删除标记。

    有关更多信息，请参阅 在 S3 存储桶中使用版本控制。

    设置 Amazon S3 以管理过期时，还应该考虑以下各项：

    - 已启用版本控制的桶

      如果当前对象版本不是删除标记，Amazon S3 将添加具有唯一的版本 ID 的删除标记。这会使当前对象版本变为非当前版本，并使删除标记变为当前版本。

- 已暂停版本控制的桶

  在已暂停版本控制的存储桶中，到期操作将使 Amazon S3 创建版本 ID 为 null 的删除标记。此删除标记会在版本层次结构中将任何对象版本替换为 null 版本 ID，从而实际上删除对象。

此外，Amazon S3 还提供可用于管理受版本控制的桶（即，启用了版本控制和暂停了版本控制的桶）中的非当前对象版本的以下操作。

- **NoncurrentVersionTransition** 操作元素 – 使用此操作可指定 Amazon S3 何时将对象转换到指定的存储类。您可以根据自对象变为非当前对象以来的一定天数来确定此到期日期。除了天数之外，还可以提供要保留的最大非当前版本数（介于 1 和 100 之间）。此值确定在 Amazon S3 可以对给定版本执行关联操作之前，必须存在多少个较新的非当前版本。Amazon S3 将转换超过要保留的指定数量的任何其它非当前版本。

  要指定最大非当前版本数量，您还必须提供 Filter 元素。如果您未指定 Filter 元素，Amazon S3 会在您提供最大数量的非当前版本时生成 InvalidRequest 错误。

  有关转换对象的更多信息，请参阅 受支持的转换和相关限制。有关当您在 NoncurrentVersionTransition 操作中指定天数时 Amazon S3 如何计算日期的详细信息，请参阅 生命周期规则：基于对象的存在期限。
- **NoncurrentVersionExpiration** 操作元素 – 使用此操作可指示 Amazon S3 何时永久删除对象的非当前版本。删除的对象无法恢复。您可以根据自对象变为非当前对象以来的一定天数来确定此到期日期。除了天数之外，还可以提供要保留的最大非当前版本数（介于 1 和 100 之间）。此值指定在 Amazon S3 可以对给定版本执行关联操作之前，必须存在多少较新的非当前版本。Amazon S3 将永久删除超过要保留的指定数量的任何其它非当前版本。

  要指定最大非当前版本数量，您还必须提供 Filter 元素。如果您未指定 Filter 元素，Amazon S3 会在您提供最大数量的非当前版本时生成 InvalidRequest 错误。

  当您需要更正任何意外删除或覆盖时，非当前对象的延时移除可能很有帮助。例如，您可以配置一个到期规则，以便在对象变为非当前版本五天后删除非当前版本。例如，假设在 2014 年 1 月 1 日上午 10:30 UTC，您创建了名为 photo.gif 的对象（版本 ID 111111）。在 2014 年 2 月 1 日上午 11:30 UTC，您意外删除了 photo.gif（版本 ID 111111），这将导致使用新版本 ID（如版本 ID 4857693）创建一个删除标记。您现在有五天时间可以在永久删除之前，恢复原始版本的 photo.gif (版本 ID 111111)。在 2014 年 1 月 8 日 00:00 UTC，有关到期的生命周期规则运行并永久删除 photo.gif（版本 ID 111111）（在它成为非当前版本五天之后）。

有关当您在 NoncurrentVersionExpiration 操作中指定天数时 Amazon S3 如何计算日期的详细信息，请参阅 生命周期规则：基于对象的存在期限。

> **Note**
>
> 对象到期生命周期配置不会移除未完成的分段上传。要移除未完成的分段上传，您必须使用 AbortIncompleteMultipartUpload 生命周期配置操作，如本节后面所述。

除了转换和到期操作之外，您还可以使用以下生命周期配置操作，来指示 Amazon S3 停止未完成的分段上传或移除到期的对象删除标记：

- **AbortIncompleteMultipartUpload** 操作元素 – 使用此元素可设置您希望允许分段上传保持运行的最长时间（天）。如果适用的分段上传（由生命周期规则中指定的键名称 prefix 确定）未在预定义的时间段内成功完成，Amazon S3 将停止未完成的分段上传。有关更多信息，请参阅 中止分段上传。

  > **Note**
  >
  > 如果规则具有使用对象标签的筛选条件，则您无法在此规则中指定此生命周期操作。

- **ExpiredObjectDeleteMarker** 操作元素 – 在启用了版本控制的存储桶中，包含零个非当前版本的删除标记称为到期对象删除标记。您可以使用此生命周期操作来指示 Amazon S3 移除到期对象移除标记。有关示例，请参阅示例 7：移除过期对象删除标记。

  > **Note**
  >
  > 如果规则具有使用对象标签的筛选条件，则您无法在此规则中指定此生命周期操作。

Amazon S3 如何计算对象已成为非当前版本的时间长度

在启用版本控制的桶中，您可以有一个对象的多个版本。当前版本始终有一个，以及零个或零个以上非当前版本。每次上传对象时，当前版本都保留为非当前版本，新添加的版本 (后继者) 会成为当前版本。为了确定对象成为非当前版本的天数，Amazon S3 会查看其后继者的创建时间。Amazon S3 使用自后继者创建以来的天数作为对象成为非当前版本的天数。

在使用 S3 生命周期配置时还原对象的以前版本

按照还原早期版本中的详细说明，您可以使用以下两种方法中的任一种来检索对象的以前版本：

- 方法 1 – 将对象的非当前版本复制到相同存储桶中。复制的对象将成为该对象的当前版本，且所有对象版本都保留。
- 方法 2 – 永久删除对象的当前版本。当您删除当前对象版本时，实际上是将非当前版本转换为该对象的当前版本。

当您正在将 S3 生命周期配置规则用于启用版本控制的存储桶时，我们建议的最佳实践是使用方法 1。
S3 生命周期在最终一致的模型下运行。在更改传播到所有 Amazon S3 系统之前，永久删除的当前版本可能不会消失。（因此，Amazon S3 可能暂时不知道此删除操作。）同时，您配置用于使非当前对象过期的生命周期规则可能会永久删除非当前对象，包括您要还原的对象。因此，复制旧版本（按照方法 1 建议）是更安全的替代方法。

生命周期操作与桶版本控制状态

生命周期规则：基于对象的存在期限

您可以指定 Amazon S3 能够执行指定操作的时间段 [自对象创建（或修改）以来的天数]。

当您在 S3 生命周期配置中的 `Transition` 和 `Expiration` 操作中指定天数时，请注意以下事项：

- 您指定的值是自对象创建以来将发生此操作的天数。
- Amazon S3 按以下方式计算时间：将在规则中指定的天数与对象创建时间相加，然后将得出的时间舍入至下一日的午夜 UTC。例如，如果对象的创建时间是 2014 年 1 月 15 日上午 10:30 UTC，并且您在转换规则中指定了 3 天，则对象的转换日期将计算为 2014 年 1 月 19 日 00:00 UTC。

Note

Amazon S3 仅为每个对象保持上次修改日期。例如，Amazon S3 控制台在对象的属性窗格中显示上次修改日期。最初创建新对象时，此日期反映对象的创建日期。如果您替换对象，此日期会相应地更改。因此，创建日期与上次修改日期同义。

当在生命周期配置中的 NoncurrentVersionTransition 和 NoncurrentVersionExpiration 操作中指定天数时，请注意以下几点：

- 您指定的值是从对象版本变为非当前版本以来（即，当对象被覆盖或删除时）的天数，作为 Amazon S3 将对指定的对象执行操作的时间。
- Amazon S3 按以下方式计算时间：将规则中指定的天数与创建对象的新后继者版本的时间相加，然后将得出的时间舍入至下一日的午夜 UTC。例如，在您的存储桶中，假设某个对象的当前版本的创建时间是 2014 年 1 月 1 日上午 10:30 UTC。如果替换当前版本的对象新版本的创建时间是 2014 年 1 月 15 日上午 10:30 UTC，并且您在转换规则中指定了 3 天，则对象的转换日期计算为 2014 年 1 月 19 日 00:00 UTC。

生命周期规则：基于特定日期

当在 S3 生命周期规则中指定操作时，您可以指定希望 Amazon S3 执行此操作的日期。到达特定日期时，Amazon S3 会向所有合格对象应用该操作（基于筛选条件）。

如果为 S3 生命周期操作指定一个过去的日期，所有合格对象会立即符合该生命周期操作的条件。

> **Important**
>
> 基于日期的操作并非一次性操作。即使过了该日期后，只要规则状态为 Enabled，Amazon S3 仍会继续应用该基于日期的操作。
> 例如，假设您指定一个基于日期的 Expiration 操作来删除所有对象（假设规则中未指定任何筛选条件）。在指定日期，Amazon S3 会使桶中的所有对象过期。此外，Amazon S3 还会继续使您在存储桶中创建的所有新对象到期。要停止生命周期操作，您必须从生命周期规则中移除操作，禁用规则或从生命周期配置中删除规则。

此日期值必须符合 ISO 8601 格式。时间始终为午夜 UTC。

> **Note**
>
> 您无法使用 Amazon S3 控制台创建基于日期的生命周期规则，但是可以查看、禁用或删除这类规则。

S3 生命周期配置的示例

此部分提供 S3 生命周期配置的示例。每个示例演示如何在各个示例方案中指定 XML。

Important

当您在 S3 生命周期配置中有多个规则时，对象可能变得符合条件可在同一天执行多个 S3 生命周期操作。在这种情况下，Amazon S3 遵循以下一般规则：

- 永久删除优先于转换。
- 转换优先于删除标记的创建。
- 当对象有资格进行 S3 Glacier Flexible Retrieval 和 S3 Standard-IA（或 S3 One Zone-IA）转换时，Amazon S3 将选择 S3 Glacier Flexible Retrieval 转换。

有关示例，请参阅示例 5：重叠的筛选条件、冲突的生命周期操作，以及 Amazon S3 如何处理非版本化的存储桶。

主题

示例 1：指定筛选条件

每个 S3 生命周期规则都包含一个筛选条件，该筛选条件可用于确定存储桶中适用 S3 生命周期规则的一部分对象。以下 S3 生命周期配置显示了如何指定筛选条件的示例。

- 在此 S3 生命周期配置规则中，筛选条件指定了一个键名称前缀 (`tax/`)。因此，此规则应用于带前缀 `tax/` 的对象，例如 `tax/doc1.txt` 和 `tax/doc2.txt`。

  此规则指定两个指示 Amazon S3 完成以下任务的操作：

  - 在对象创建 365 天（一年）后将其转换为 S3 Glacier Flexible Retrieval 存储类。

- 在对象创建 3650 天（10 年）后将其删除（Expiration 操作）。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition and Expiration Rule</ID>
    <Filter>
       <Prefix>tax/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

您可以为每个操作指定一个日期，而不用以创建后的天数的形式指定对象期限。但是，您不能在同一规则中同时使用 Date 和 Days。

- 如果希望 S3 生命周期规则应用于存储桶中的所有对象，请指定一个空前缀。在下面的配置中，该规则指定了一个 Transition 操作，该操作将指示 Amazon S3 在创建后 0 天将对象转移到 S3 Glacier Flexible Retrieval 存储类。此规则意味着对象在创建后有资格在世界标准时间午夜归档到 S3 Glacier Flexible Retrieval。有关生命周期约束的更多信息，请参阅 约束。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all object same-day upon creation</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

- 您可以在筛选条件中指定零或一个前缀以及零或多个对象标签。以下示例代码将 S3 生命周期规则应用于带 tax/ 前缀的一部分对象以及包含具有特定键和值的两个标签的对象。在指定多个筛选条件时，您必须按下面所示包含 <And>（Amazon S3 会应用逻辑 AND 以将指定筛选条件组合起来）。

```
...
<Filter>
   <And>
      <Prefix>tax/</Prefix>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
         <Value>value2</Value>
      </Tag>
    </And>
</Filter>
...
```

> **Note**
>
> 如果您有一个或多个前缀以相同的字符开头，则可以通过在筛选条件中指定不带尾部斜杠 (/) 的部分前缀，来将所有这些前缀包含在规则中。例如，假设您有以下前缀：
>
> ```
> sales1999/
> sales2000/
> sales2001/
> ```
>
> 要在规则中包含所有三个前缀，请在生命周期规则中指定 <Prefix>sales</Prefix>。

- 您可以仅基于标签筛选对象，而不使用前缀。例如，以下 S3 生命周期规则应用于具有两个指定标签的对象（该规则未指定任何前缀）。

```
...
<Filter>
   <And>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
```

```
      <Tag>
         <Key>key2</Key>
         <Value>value2</Value>
      </Tag>
    </And>
</Filter>
...
```

如果要从生命周期规则中排除 特定前缀，请使用标签来标记要包含 在规则内的前缀中的所有对象。

示例 2：禁用生命周期规则

您可以临时禁用 S3 生命周期规则。以下 S3 生命周期配置指定了两个规则：

- 规则 1 指示 Amazon S3 在带 `logs/` 前缀的对象创建后立即将其转换为 S3 Glacier Flexible Retrieval 存储类。
- 规则 2 指示 Amazon S3 在带 `documents/` 前缀的对象创建后立即将其转换为 S3 Glacier Flexible Retrieval 存储类。

在此配置中，启用了规则 1 但禁用了规则 2。Amazon S3 忽略禁用的规则。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule2</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Disabled</Status>
    <Transition>
      <Days>0</Days>
```

```
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

示例 3：在对象的生命周期内逐步将存储类降级

在本示例中，您使用 S3 生命周期配置在对象的生命周期内逐步将存储类降级。级别降低可帮助减少存储成本。有关定价的更多信息，请参阅 Amazon S3 定价。

以下 S3 生命周期配置指定了应用于带键名前缀 `logs/` 的对象的规则。该规则指定了以下操作：

- 两个转换操作：
  - 在对象创建 30 天后将其转换为 S3 Standard-IA 存储类。
  - 在对象创建 90 天后将其转换为 S3 Glacier Flexible Retrieval 存储类。
- 一个过期操作，指示 Amazon S3 在对象创建一年后将其删除。

```
<LifecycleConfiguration>
  <Rule>
    <ID>example-id</ID>
    <Filter>
       <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>90</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> 如果所有 S3 生命周期操作都应用于同一组对象（由筛选条件标识），您可以使用一个规则来描述这些操作。或者，您也可以添加多个规则，每个规则指定一个不同的筛选条件。

> **Important**
>
> 当您在 S3 生命周期配置中有多个规则时，对象可能变得符合条件可在同一天执行多个 S3 生命周期操作。在这种情况下，Amazon S3 遵循以下一般规则：
>
> - 永久删除优先于转换。
> - 转换优先于删除标记的创建。
> - 当对象有资格进行 S3 Glacier Flexible Retrieval 和 S3 Standard-IA（或 S3 One Zone-IA）转换时，Amazon S3 将选择 S3 Glacier Flexible Retrieval 转换。
>
> 有关示例，请参阅示例 5：重叠的筛选条件、冲突的生命周期操作，以及 Amazon S3 如何处理非版本化的存储桶。

示例 4：指定多个规则

如果您希望不同的对象有不同的 S3 生命周期操作，则可以指定多个规则。以下 S3 生命周期配置有两个规则：

- 规则 1 应用于带键名前缀 `classA/` 的对象。它指示 Amazon S3 在对象创建一年后将其转换为 S3 Glacier Flexible Retrieval 存储类，并在对象创建 10 年后使它们过期。
- 规则 2 应用于带键名前缀 `classB/` 的对象。它指示 Amazon S3 在对象创建 90 天后将其转换为 S3 Standard-IA 存储类，并在对象创建 1 年后将其删除。

```
<LifecycleConfiguration>
    <Rule>
        <ID>ClassADocRule</ID>
        <Filter>
           <Prefix>classA/</Prefix>
        </Filter>
```

```
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>ClassBDocRule</ID>
        <Filter>
            <Prefix>classB/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <Expiration>
             <Days>365</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

Important

当您在 S3 生命周期配置中有多个规则时，对象可能变得符合条件可在同一天执行多个 S3 生命周期操作。在这种情况下，Amazon S3 遵循以下一般规则：

- 永久删除优先于转换。
- 转换优先于删除标记的创建。
- 当对象有资格进行 S3 Glacier Flexible Retrieval 和 S3 Standard-IA（或 S3 One Zone-IA）转换时，Amazon S3 将选择 S3 Glacier Flexible Retrieval 转换。

有关示例，请参阅示例 5：重叠的筛选条件、冲突的生命周期操作，以及 Amazon S3 如何处理非版本化的存储桶。

示例 5：重叠的筛选条件、冲突的生命周期操作，以及 Amazon S3 如何处理非版本化的存储桶

您可能会指定一个在其中指定了重叠的前缀或操作的 S3 生命周期配置。

通常，S3 生命周期将针对成本进行优化。例如，如果两个过期策略重叠，将采用较短的过期策略，以便数据存储不会超过预期时间。同样，如果两个转换策略重叠，S3 生命周期将对象转换为较低成本的存储类。

在这两种情况下，S3 生命周期尝试选择最便宜的路径。该一般规则的一个例外情况是 S3 Intelligent-Tiering 存储类。S3 生命周期希望采用 S3 Intelligent-Tiering 而不是任何其他存储类，但 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类除外。

以下示例说明 Amazon S3 如何解决潜在的冲突。

Example 1：重叠的前缀（无冲突）

以下示例配置包含两个规则，它们指定了如下所示的重叠前缀：

- 第一个规则指定了一个空筛选条件，指示存储桶中的所有对象。
- 第二个规则指定了一个键名前缀 (`logs/`)，指示仅一部分对象。

规则 1 请求 Amazon S3 在所有对象创建一年后删除这些对象。规则 2 请求 Amazon S3 在创建 30 天后将对象子集转换为 S3 Standard-IA 存储类。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
```

```
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>30</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

由于在这种情况下没有冲突，因此 Amazon S3 将在创建 30 天后将带有 `logs/` 前缀的对象转换为 S3 Standard-IA 存储类。当任何对象在创建一年后到达时，它将被删除。

Example 2：冲突的生命周期操作

本示例配置中有两个规则，它们指示 Amazon S3 在对象的生命周期中同时对同一组对象执行两个不同的操作：

- 两个规则指定了相同的键名前缀，因此两个规则都应用于同一组对象。
- 当应用两个规则时，它们指定了相同的“对象创建后的 365 天”。
- 一个规则指示 Amazon S3 将对象转换为 S3 Standard-IA 存储类，另一个规则希望 Amazon S3 使对象同时过期。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
```

```
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

在这种情况下，由于您希望对象过期（要移除），因此在更改存储类时没有意义，因此 Amazon S3 在这些对象上选择过期操作。

Example 3：导致冲突的生命周期操作的重叠的前缀

在本示例中，配置包含两个指定重叠前缀的规则，如下所示：

- 规则 1 指定了一个空前缀 (指示所有对象)。
- 规则 2 指定了一个键名前缀 (`logs/`)，用于确定所有对象中的一部分。

对于带 `logs/` 键名前缀的一部分对象，两个规则中的 S3 生命周期操作都适用。一个规则指示 Amazon S3 在对象创建 10 天后对其进行转换，另一个规则指示 Amazon S3 在对象创建 365 天后对其进行转换。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>10</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA<StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
```

```
</LifecycleConfiguration>
```

在这种情况下，Amazon S3 将选择在对象创建 10 天后转换它们。

Example 4：基于标签的筛选和随之出现的冲突的生命周期操作

假设您有包含两个规则（每个规则各指定一个标签筛选条件）的以下 S3 生命周期配置：

- 规则 1 指定了基于标签的筛选条件 (`tag1/value1`)。此规则指示 Amazon S3 在对象创建 365 天后将其转换为 S3 Glacier Flexible Retrieval 存储类。
- 规则 2 指定了基于标签的筛选条件 (`tag2/value2`)。此规则指示 Amazon S3 在对象创建 14 天后使其过期。

S3 生命周期配置如下所示。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Tag>
         <Key>tag1</Key>
         <Value>value1</Value>
      </Tag>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>GLACIER<StorageClass>
      <Days>365</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Tag>
         <Key>tag2</Key>
         <Value>value2</Value>
      </Tag>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>14</Days>
    </Expiration>
```

```
   </Rule>
</LifecycleConfiguration>
```

如果对象有两个标签，那么 Amazon S3 必须决定要遵循哪个规则。在这种情况下，Amazon S3 将在对象创建 14 天后使其过期。该对象将被删除，因此转换操作不会起作用。

> **Important**
>
> 当您在 S3 生命周期配置中有多个规则时，对象可能变得符合条件可在同一天执行多个 S3 生命周期操作。在这种情况下，Amazon S3 遵循以下一般规则：
>
> - 永久删除优先于转换。
> - 转换优先于删除标记的创建。
> - 当对象有资格进行 S3 Glacier Flexible Retrieval 和 S3 Standard-IA（或 S3 One Zone-IA）转换时，Amazon S3 将选择 S3 Glacier Flexible Retrieval 转换。
>
> 有关示例，请参阅示例 5：重叠的筛选条件、冲突的生命周期操作，以及 Amazon S3 如何处理非版本化的存储桶。

示例 6：为启用了版本控制的存储桶指定生命周期规则

假设您有一个启用了版本控制的存储桶，这意味着对于每个对象，您都有一个当前版本以及零个或零个以上的非当前版本。（有关 S3 版本控制的更多信息，请参阅 在 S3 存储桶中使用版本控制。）在此示例中，您想要保留一年的历史记录，然后删除非当前版本。S3 生命周期配置支持保留任何对象的 1 至 100 个版本。

为了节省存储开销，您希望在非现行的版本成为非现行（假设这些非现行的对象是不需要实时访问的冷数据）后 30 天将非现行的版本移至 S3 Glacier Flexible Retrieval。此外，您还期望在对象创建 90 天后降低当前版本的访问频率，因此您可能会选择将这些对象移动到 S3 Standard-IA 存储类。

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix></Prefix>
        </Filter>
```

```
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <NoncurrentVersionTransition>
            <NoncurrentDays>30</NoncurrentDays>
            <StorageClass>GLACIER</StorageClass>
        </NoncurrentVersionTransition>
       <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>5</NewerNoncurrentVersions>
            <NoncurrentDays>365</NoncurrentDays>
       </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

示例 7：移除过期对象删除标记

已启用版本控制的存储桶的每个对象有一个当前版本，以及零个或多个非当前版本。在删除对象时，请注意以下几点：

- 如果您在删除请求中未指定版本 ID，Amazon S3 将添加一个删除标记，而不是删除对象。当前对象版本将变为非当前版本，然后删除标记将变为当前版本。
- 如果您在删除请求中指定了版本 ID，Amazon S3 将永久删除对象版本（不会创建删除标记）。
- 包含零个非当前版本的删除标记称为过期对象删除标记。

此示例演示了可在存储桶中创建过期对象删除标记的场景，以及如何使用 S3 生命周期配置指示 Amazon S3 删除过期对象删除标记。

假设您编写了一个 S3 生命周期配置，该配置使用 `NoncurrentVersionExpiration` 操作以在对象变为非当前版本 30 天之后移除这些非当前版本，并最多保留 10 个非当前版本，如下面的示例所示。

```
<LifecycleConfiguration>
    <Rule>
        ...
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
```

```
</LifecycleConfiguration>
```

NoncurrentVersionExpression 操作不应用于当前的对象版本。它只会移除非当前版本。

对于当前对象版本，您可以根据当前对象版本是否遵循了明确定义的生命周期，通过以下方式管理其生命周期：

- 当前对象版本遵循明确定义的生命周期。

  在这种情况下，您可以将 S3 生命周期配置与 Expiration 操作结合使用，以指示 Amazon S3 移除当前版本，如以下示例所示。

```
<LifecycleConfiguration>
    <Rule>
        ...
        <Expiration>
           <Days>60</Days>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

  在此示例中，Amazon S3 将在每个当前对象版本创建 60 天之后通过为其添加删除标记来删除它们。这过程会使当前对象版本变为非当前版本，并使删除标记变为当前版本。有关更多信息，请参阅在 S3 存储桶中使用版本控制。

> **Note**
>
> 不能在同一规则中同时指定 Days 和 ExpiredObjectDeleteMarker 标签。当您指定 Days 标记时，Amazon S3 将于删除标记过旧而无法满足年限条件时自动执行 ExpiredObjectDeleteMarker 清理。要在删除标记成为唯一版本时立刻清理，只能使用 ExpiredObjectDeleteMarker 标签创建单独的规则。

同一 S3 生命周期配置中的 NoncurrentVersionExpiration 操作将在对象变为非当前对象 30 天后删除它们。因此，在此示例中，所有对象版本在对象创建 90 天后被永久删除。虽然您可以在此过程中创建过期的对象删除标记，但 Amazon S3 会为您检测并移除过期对象删除标记。

- 当前对象版本没有明确定义的生命周期。

  在这种情况下，您可以在不需要对象时手动删除它们，从而创建具有一个或多个非当前版本的删除标记。如果使用 NoncurrentVersionExpiration 操作的 S3 生命周期配置删除了所有非当前版本，则您现在会获得过期对象删除标记。

  特别针对这种情况，S3 生命周期配置提供了 Expiration 操作，让您可以请求移除过期对象删除标记。

```
<LifecycleConfiguration>
    <Rule>
       <ID>Rule 1</ID>
        <Filter>
          <Prefix>logs/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Expiration>
           <ExpiredObjectDeleteMarker>true</ExpiredObjectDeleteMarker>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

通过在 Expiration 操作中将 ExpiredObjectDeleteMarker 元素设为 true，您可以指示 Amazon S3 移除过期对象删除标记。

> **Note**
>
> 当您使用 ExpiredObjectDeleteMarker S3 生命周期操作时，该规则无法指定基于标签的筛选条件。

示例 8：用于中止分段上传的生命周期配置

您可以使用 Amazon S3 的分段上传 REST API 操作来在分段中上载大型对象。有关分段上传的更多信息，请参阅 使用分段上传来上传和复制对象。

通过使用 S3 生命周期配置，当分段上传在指定的启动后的天数内未完成时，您可以指示 Amazon S3 停止未完成的分段上传（通过在规则中指定的键名称前缀确定）。当 Amazon S3 中止分段上传时，它将删除与分段上传关联的所有分段。此流程通过确保您没有包含存储在 Amazon S3 中的分段的未完成分段上传，从而帮助控制存储成本。

> **Note**
>
> 当您使用 AbortIncompleteMultipartUpload S3 生命周期操作时，该规则无法指定基于标签的筛选条件。

下面是使用 AbortIncompleteMultipartUpload 操作指定规则的示例 S3 生命周期配置。此操作将指引 Amazon S3 在分段上传启动 7 天后停止未完成的分段上传。

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix>SomeKeyPrefix/</Prefix>
        </Filter>
        <Status>rule-status</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

例 9：使用基于大小的规则进行生命周期

您可以创建仅根据对象的大小转换对象的规则。您可以指定最小大小（ObjectSizeGreaterThan）或最大大小（ObjectSizeLessThan），也可以指定一个对象大小范围，单位为字节。当使用多个筛选条件（如前缀和大小规则）时，必须将筛选条件包装在 <And> 元素中。

> **Note**
>
> ObjectSizeGreaterThan 和 ObjectSizeLessThan 筛选条件不包含所指定的值。有关更多信息，请参阅 the section called “Filter 元素”。

```
<LifecycleConfiguration>
```

```
  <Rule>
    <ID>Transition with a prefix and based on size</ID>
    <Filter>
       <And>
          <Prefix>tax/</Prefix>
          <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
       </And>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

当通过同时使用 ObjectSizeGreaterThan 和 ObjectSizeLessThan 元素指定范围时，最大对象大小必须大于最小对象大小。当使用多个筛选条件时，必须将筛选条件包装在 <And> 元素中。以下示例显示了如何指定范围在 500 字节到 64000 字节之间的对象。

```
<LifecycleConfiguration>
    <Rule>
        ...
          <And>
             <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
             <ObjectSizeLessThan>64000</ObjectSizeLessThan>
          </And>
    </Rule>
</LifecycleConfiguration>
```

您还可以创建规则，专门使没有数据的非当前对象过期，包括在启用版本控制的存储桶中创建的非当前删除标记对象。以下示例使用 NoncurrentVersionExpiration 操作，在对象变为非当前版本 30 天之后删除这些非当前版本，并最多保留 10 个非当前的对象版本。它还使用该 ObjectSizeLessThan 元素只筛选没有数据的对象。

```
<LifecycleConfiguration>
  <Rule>
    <ID>Expire noncurrent with size less than 1 byte</ID>
    <Filter>
       <ObjectSizeLessThan>1</ObjectSizeLessThan>
    </Filter>
    <Status>Enabled</Status>
```

```
    <NoncurrentVersionExpiration>
       <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
       <NoncurrentDays>30</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

Amazon S3 清单

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

您可以使用 Amazon S3 清单来帮助管理您的存储。例如，您可以出于业务、合规性和法规需要，使用该清单来审核和报告对象的复制和加密状态。您还可以使用 Amazon S3 清单简化和加快业务工作流和大数据任务，此清单可以有计划地取代 Amazon S3 同步 `List` API 操作。Amazon S3 清单不使用 `List` API 操作来审计对象且不会影响存储桶的请求速率。

Amazon S3 清单每天或每周为 S3 存储桶或具有共享前缀的对象（即，其名称以通用字符串开头的对象），提供用于列出您的对象及其对应元数据的逗号分隔值（CSV）、Apache 优化的行列式（ORC）或 Apache Parquet 输出文件。如果您设置每周清单，则在初始报告后的每个星期日（UTC 时区）生成一份报告。有关 Amazon S3 清单定价的信息，请参阅 Amazon S3 定价。

您可以为存储桶配置多个清单列表。配置清单列表时，可以指定以下内容：

- 清单中应包含哪些对象元数据
- 是列出所有对象版本，还是仅列出当前版本
- 在哪里存储清单列表文件输出
- 是每天生成清单，还是每周生成清单
- 是否加密清单列表文件

您可以通过 Amazon Athena、Amazon Redshift Spectrum 以及其他工具（如 Presto、Apache Hive 和 Apache Spark）使用标准 SQL 查询 Amazon S3 清单。有关使用 Athena 查询清单文件的更多信息，请参阅the section called “使用 Athena 查询清单”。

源存储桶和目标存储桶

由清单列出其对象的存储桶称为源存储桶。在其中存储清单列表文件的存储桶称为目标存储桶。

源存储桶

清单列出了源存储桶中存储的对象。您可以获得整个存储桶的清单列表，也可以通过对象键名称前缀筛选此列表。

源存储桶：

- 包含在清单中列出的对象
- 包含清单的配置

目标存储桶

Amazon S3 清单列表文件将写入目标存储桶。要对目标存储桶内公共位置中的所有清单列表文件进行分组，您可以在清单配置中指定目标前缀。

目标存储桶：

- 包含清单文件列表。
- 包含清单文件，其中列出了存储在目标存储桶中的所有清单列表文件。有关更多信息，请参阅 清单 Manifest。
- 必须具有向 Amazon S3 授予验证存储桶的所有权的权限和将文件写入存储桶的权限的存储桶策略。
- 必须与源存储桶位于同一 AWS 区域。
- 可以与源存储桶相同。
- 可以由与拥有源存储桶的账户不同的 AWS 账户 拥有。

Amazon S3 清单列表

清单列表文件包含源存储桶中对象的列表以及每个对象的元数据。清单列表文件按以下格式之一存储在目标存储桶中：

- 作为使用 GZIP 压缩的 CSV 文件
- 作为使用 ZLIB 压缩的 Apache 优化行列式（ORC）文件
- 作为使用 Snappy 压缩的 Apache Parquet 文件

> **Note**
>
> 不能保证会以任何顺序对 Amazon S3 清单报告中的对象排序。

清单列表文件包含源存储桶中对象的列表以及每个所列对象的元数据：

- Bucket name (存储桶名称) – 清单所针对的存储桶的名称。
- 键名称 – 唯一地标识存储桶中的对象的对象键名（或键）。当您使用 CSV 文件格式时，键名称采用 URL 编码形式，必须解码，然后才能使用。
- 版本 ID – 对象版本 ID。在存储桶上启用版本控制后，Amazon S3 会为添加到存储桶的对象指定版本号。有关更多信息，请参阅 在 S3 存储桶中使用版本控制。（如果仅为对象的当前版本配置列表，则不包括此字段。）
- IsLatest – 如果对象的版本为最新，则设置为 `True`。（如果仅为对象的当前版本配置列表，则不包括此字段。）
- 删除标记 – 如果对象是删除标记，则设置为 `True`。有关更多信息，请参阅 在 S3 存储桶中使用版本控制。（如果您已将您的报告配置为包含您的对象的所有版本，则此字段将自动添加到报告）。
- 大小 - 以字节为单位的对象大小，不包括未完成的分段上传、对象元数据和删除标记的大小。
- Last modified date (上次修改日期) – 对象创建日期或上次修改日期（以较晚者为准）。
- ETag – 实体标签（ETag）是对象的哈希。ETag 仅反映对于对象的内容的更改，而不反映对于对象的元数据的更改。ETag 可能是对象数据的 MD5 摘要。是与不是取决于对象的创建方式和加密方式。
- 存储类 – 用于存储对象的存储类。设置为 `STANDARD`、`REDUCED_REDUNDANCY`、`STANDARD_IA`、`ONEZONE_IA`、`INTELLIGENT_TIERING`、`GLACIER` 或 `SNOW`。有关更多信息，请参阅 使用 Amazon S3 存储类。
- Multipart upload flag (分段上传标记) – 如果对象以分段上传形式上传，则设置为 `True`。有关更多信息，请参阅 使用分段上传来上传和复制对象。
- 复制状态 – 设置为 `PENDING`、`COMPLETED`、`FAILED` 或 `REPLICA`。有关更多信息，请参阅 获取复制状态信息。

- 加密状态 – 服务器端加密状态，具体取决于使用哪种加密密钥：采用 Amazon S3 托管式密钥的服务器端加密（SSE-S3）、采用 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）、采用 AWS KMS 密钥的双层服务器端加密（DSSE-C）或采用客户提供的密钥的服务器端加密（SSE-C）。设置为 `SSE-S3`、`SSE-KMS`、`DSSE-KMS`、`SSE-C` 或 `NOT-SSE`。`NOT-SSE` 状态表示对象未使用服务器端加密进行加密。有关更多信息，请参阅 利用加密来保护数据。
- S3 对象锁定保留截止日期 – 在此日期之前无法删除锁定的对象。有关更多信息，请参阅 使用 S3 对象锁定。
- S3 对象锁定保留模式 – 对于已锁定的对象，设置为 `Governance` 或 `Compliance`。有关更多信息，请参阅 使用 S3 对象锁定。
- S3 对象锁定依法保留状态 – 如果依法保留已应用于对象，则设置为 `On`。否则，其设置为 `Off`。有关更多信息，请参阅 使用 S3 对象锁定。
- S3 Intelligent-Tiering 访问层 – 如果对象存储在 S3 Intelligent-Tiering 存储类中，则为对象的访问层（频繁或不频繁）。设置为 `FREQUENT`、`INFREQUENT`、`ARCHIVE_INSTANT_ACCESS`、`ARCHIVE` 或 `DEEP_ARCHIVE`。有关更多信息，请参阅 用于自动优化访问模式不断变化或未知的数据的存储类。
- S3 存储桶密钥状态 – 设置为 `ENABLED` 或者 `DISABLED`。指示对象是否将 S3 存储桶密钥用于 SSE-KMS。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥。
- 校验和算法 – 表示用于创建对象的校验和的算法。
- 对象访问控制列表 – 每个对象的访问控制列表（ACL），用于定义哪个 AWS 账户或组被授予对此对象的访问权限以及授予的访问权限类型。“对象 ACL”字段以 JSON 格式定义。S3 清单报告包括与源存储桶中的对象关联的 ACL，即使已为该存储桶禁用了 ACL 仍会包括。有关更多信息，请参阅使用“对象 ACL”字段 和访问控制列表 (ACL) 概述。

> **Note**
>
> “对象 ACL”字段以 JSON 格式定义。清单报告将“对象 ACL”字段的值显示为以 base64 编码的字符串。
>
> 例如，假设您有以下采用 JSON 格式的“对象 ACL”字段：

```
{
        "version": "2022-11-10",
        "status": "AVAILABLE",
        "grants": [{
            "canonicalId": "example-canonical-user-ID",
            "type": "CanonicalUser",
            "permission": "READ"
        }]
```

```
}
```

“对象 ACL”字段经过编码并显示为以下以 base64 编码的字符串：

```
eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2Fs
```

要以 JSON 格式获取“对象 ACL”字段的解码值，可以在 Amazon Athena 中查询此字段。有关示例查询，请参阅使用 Amazon Athena 查询 Amazon S3 清单。

- 对象拥有者 – 对象的拥有者。

> **Note**
>
> 在对象根据其生命周期配置的生存期结束后，Amazon S3 会将该对象加入移除队列并异步移除它。因此，过期日期和 Amazon S3 删除对象的日期之间可能会有一段延迟。清单报告包括已过期但尚未移除的对象。有关 S3 生命周期中的过期操作的更多信息，请参阅即将过期的对象。

我们建议创建一个删除旧清单列表的生命周期策略。有关更多信息，请参阅 管理存储生命周期。

`s3:PutInventoryConfiguration` 权限允许用户在配置清单列表时选择之前为每个对象列出的所有元数据字段，也可以指定存储清单的目标存储桶。对目标存储桶中的对象具有读取权限的用户可以访问清单列表中提供的所有对象元数据字段。要限制对于清单报告的访问权限，请参阅向 S3 清单和 S3 分析功能授予权限。

清单一致性

并非所有对象都会显示每个清单列表中。清单列表提供了（新对象和覆盖）的 PUT 请求的最终一致性，并提供了 DELETE 请求的最终一致性。存储桶的每个清单列表都是存储桶项目的快照。这些列表最终是一致的（也就是说，列表可能不包括最近添加或删除的对象）。

要在对于对象执行操作之前验证对象的状态，我们建议您执行 `HeadObject` REST API 请求以检索对象的元数据，或在 Amazon S3 控制台中检查对象的属性。您还可以使用 AWS CLI 或 AWS SDK 检查对象元数据。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 `HeadObject`。

有关使用 Amazon S3 清单的更多信息，请参阅以下主题。

主题

配置 Amazon S3 清单

Amazon S3 清单每天根据您定义的时间表，提供您的对象和元数据的平面文件列表。您可以使用 S3 清单作为 Amazon S3 同步 `List` API 操作的计划替代方案。S3 清单提供逗号分隔值（CSV）、经过 Apache 优化的行列式（ORC）或 Apache Parquet (Parquet) 输出文件，这些文件列出了对象及其相应的元数据。

您可以将 S3 清单配置为每天或每周为 S3 存储桶或为共享前缀的对象（名称以相同字符串开头的对象）创建清单列表。有关更多信息，请参阅 Amazon S3 清单。

本部分介绍了如何设置清单，包括有关清单源存储桶和目标存储桶的详细信息。

主题

概述

Amazon S3 清单将帮助您按预定的计划创建 S3 存储桶中的对象的列表，由此管理存储。您可以为桶配置多个清单列表。清单列表将发布到目标存储桶中的 CSV、ORC 或 Parquet 文件。

设置清单的最简单方法是使用 Amazon S3 控制台，不过您也可以使用 Amazon S3 REST API、AWS Command Line Interface (AWS CLI) 或 AWS SDK。控制台将为您执行以下过程的第一步：将存储桶策略添加到目标存储桶。

为 S3 存储桶设置 Amazon S3 清单

1. 为目标存储桶添加存储桶策略。

   您必须在目标存储桶上创建存储桶策略，以向 Amazon S3 授予将对象写入已定义位置的存储桶的权限。有关策略示例，请参阅向 S3 清单和 S3 分析功能授予权限。

2. 配置一个清单以列出源存储桶中的对象并将该列表发布到目标存储桶。

   当您为源存储桶配置清单列表时，您应指定用于存储该列表的目标存储桶，以及是每天还是每周生成一次列表。您还可以配置是列出所有对象版本还是仅列出当前版本，以及要包含哪些对象元数据。

   S3 清单报告配置中的某些对象元数据字段是可选的，这意味着它们在默认情况下是可用的，但是当您向用户授予 `s3:PutInventoryConfiguration` 权限时，它们可能会受到限制。您可以使用 `s3:InventoryAccessibleOptionalFields` 条件键控制用户能否在报告中包含这些可选元数据字段。

   有关 S3 清单中可用的可选元数据字段的更多信息，请参阅《Amazon Simple Storage Service API Reference》中的 OptionalFields。有关在清单配置中限制对某些可选元数据字段的访问的更多信息，请参阅控制 S3 清单报告配置的创建。

   您可以指定使用具有 Amazon S3 托管式密钥 (SSE-S3) 或具有 AWS Key Management Service (AWS KMS) 客户自主管理型密钥（SSE-KMS）的服务器端加密来加密清单列表文件。

   > **Note**
   >
   > 使用 S3 清单进行的 SSE-KMS 加密不支持 AWS 托管式密钥 (`aws/s3`)。

   有关 SSE-S3 和 SSE-KMS 的更多信息，请参阅 使用服务器端加密保护数据。如果您打算使用 SSE-KMS 加密，请参阅步骤 3。

   - 有关如何使用控制台配置清单列表的信息，请参阅使用 S3 控制台配置清单。
   - 要使用 Amazon S3 API 配置清单列表，请使用 PutBucketInventoryConfiguration REST API 操作，或者使用 AWS CLI 或 AWS SDK 中的等效操作。

3. 要使用 SSE-KMS 加密清单列表文件，授予 Amazon S3 使用 AWS KMS key 的权限。

   您可以使用 Amazon S3 控制台、Amazon S3 REST API、AWS CLI 或 AWS SDK，为清单列表文件配置加密。无论选择哪种方式，您都必须授予 Amazon S3 相应权限使用客户托管密钥来加密清单

文件。可通过修改要用于加密清单文件的客户托管密钥的密钥策略，向 Amazon S3 授予权限。有关更多信息，请参阅 向 Amazon S3 授予权限以使用客户自主管理型密钥进行加密。

存储清单列表文件的目标存储桶可以由与拥有源存储桶的账户不同的 AWS 账户拥有。如果您使用 SSE-KMS 加密功能执行 Amazon S3 清单跨账户操作，则在配置 S3 清单时，建议您使用完全限定的 KMS 密钥 ARN。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的使用 SSE-KMS 加密进行跨账户操作和 ServerSideEncryptionByDefault。

创建目标存储桶策略

如果您通过 Amazon S3 控制台创建清单配置，Amazon S3 会在目标存储桶上自动创建一个存储桶策略，以授予 Amazon S3 对存储桶的写入权限。但是，如果您通过 AWS CLI、AWS SDK 或 Amazon S3 REST API 创建清单配置，则必须在目标存储桶上手动添加存储桶策略。有关更多信息，请参阅 向 S3 清单和 S3 分析功能授予权限。S3 清单目标存储桶策略允许 Amazon S3 将清单报告数据写入存储桶。

如果在您尝试创建存储桶策略出现错误，则将为您提供相关修复说明。例如，如果您选择了其他 AWS 账户 中的目标存储桶，而没有权限读取和写入存储桶策略，则您会看到一条错误消息。

在这种情况下，目标存储桶拥有者必须将存储桶策略添加到目标存储桶中。如果策略未添加到目标存储桶中，则您不会获得清单报告，因为 Amazon S3 无权写入目标存储桶。如果源桶属于其他账户而非当前用户的账户，则必须在策略中替换源桶拥有者的正确账户 ID。

向 Amazon S3 授予权限以使用客户自主管理型密钥进行加密

要向 Amazon S3 授予使用 AWS Key Management Service（AWS KMS）客户自主管理型密钥进行服务器端加密的权限，必须使用密钥策略。要更新密钥策略，以便您能够使用客户自主管理型密钥，请使用以下过程。

授予 Amazon S3 使用客户自主管理型密钥进行加密的权限

1. 使用拥有客户自主管理型密钥的 AWS Management Console，登录 AWS 账户。
2. 从 https://console.aws.amazon.com/kms 打开 AWS KMS 控制台。
3. 要更改 AWS 区域，请使用页面右上角的区域选择器。
4. 在左侧导航窗格中，选择 Customer managed keys (客户托管密钥)。
5. 在客户自主管理型密钥下，选择要用于加密清单文件的客户托管密钥。
6. 在 Key Policy (密钥策略) 部分中，选择 Switch to policy view (切换到策略视图)。

7. 要更新密钥策略，选择 Edit (编辑)。
8. 在编辑密钥策略页上，将以下各行添加到现有密钥策略。对于 *source-account-id* 和 *amzn-s3-demo-source-bucket*，为您的用例提供适当的值。

```
{
    "Sid": "Allow Amazon S3 use of the customer managed key",
    "Effect": "Allow",
    "Principal": {
        "Service": "s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition":{
      "StringEquals":{
         "aws:SourceAccount":"source-account-id"
     },
      "ArnLike":{
        "aws:SourceARN": "arn:aws:s3:::amzn-s3-demo-source-bucket"
     }
   }
}
```

9. 选择 Save changes（保存更改）。

有关创建客户托管密钥和使用密钥策略的更多信息，请参阅 AWS Key Management Service 开发人员指南中的以下链接：

- 管理密钥
- AWS KMS 中的密钥策略

使用 S3 控制台配置清单

按照以下说明使用 S3 控制台配置清单。

> **Note**
>
> Amazon S3 交付第一份清单报告可能需要长达 48 小时。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。在桶列表中，选择要为其配置 Amazon S3 清单的桶的名称。
3. 选择管理选项卡。
4. 在 Inventory configurations (清单配置) 下，选择 Create inventory configuration (创建清单配置)。
5. 对于清单配置名称，输入名称。
6. 对于清单范围，请执行以下操作：
   - 输入可选的前缀。
   - 选择要包括的对象版本：仅当前版本或包括所有版本。
7. 在 Report details (报告详细信息) 下，选择要将报告保存到的 AWS 账户 的位置：此账户或另一个账户。
8. 在目标下，选择要将清单报告保存到的目标存储桶。

   目标存储桶必须位于与您为其设置清单的桶相同的 AWS 区域 中。目标存储桶可处于不同的 AWS 账户 中。指定目标存储桶时，您还可以添加可选前缀，以将清单报告组合在一起。

   在目标存储桶字段下，您将看到目标存储桶权限声明，该声明添加到目标存储桶策略中以允许 Amazon S3 在该桶中放置数据。有关更多信息，请参阅 创建目标存储桶策略。
9. 在频率下，选择生成报告的频率：每日或每周。
10. 对于输出格式，为报告选择以下格式之一：
    - CSV – 如果您计划将此清单报告用于 S3 批量操作，或者如果您想在其他工具（例如 Microsoft Excel）中分析此报告，请选择 CSV。
    - Apache ORC
    - Apache Parquet
11. 在 Status (状态) 下，选择 Enable (启用) 或 Disable (禁用)。
12. 要配置服务器端加密，请在清单报告加密下执行以下步骤：

    a. 在服务器端加密下，选择不要指定加密密钥或指定加密密钥来加密数据。

       - 要在将对象存储在 Amazon S3 中时保留适用于对象原定设置服务器端加密的桶设置，请选择不要指定加密密钥。只要桶目标启用了 S3 存储桶密钥，复制操作就将在目标存储桶上应用 S3 存储桶密钥。

> **Note**
>
> 如果指定目标的存储桶策略要求在将对象存储到 Amazon S3 之前对其进行加密，则必须指定加密密钥。否则，将对象复制到目标将失败。

- 要在将对象存储到 Amazon S3 之前对其进行加密，请选择指定加密密钥。

b. 如果您选择指定加密密钥，那么在加密类型下，您必须选择 Amazon S3 托管密钥（SSE-S3）或 AWS Key Management Service 密钥（SSE-KMS）。

SSE-S3 使用最强的数据块密码之一 [即 256 位高级加密标准（AES-256）] 来加密每个对象。SSE-KMS 为您提供了对密钥的更多控制。有关 SSE-S3 的更多信息，请参阅使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。有关 SSE-KMS 的更多信息，请参阅使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。

> **Note**
>
> 要使用 SSE-KMS 加密清单列表文件，必须授予 Amazon S3 使用客户自主管理型密钥的权限。有关说明，请参阅授予 Amazon S3 使用 KMS 密钥进行加密的权限。

c. 如果您选择了 AWS Key Management Service 密钥（SSE-KMS），则在 AWS KMS key 之下，您可以通过以下选项之一指定 AWS KMS 密钥。

> **Note**
>
> 如果存储清单列表文件的目标存储桶由其他 AWS 账户拥有，请确保使用完全限定的 KMS 密钥 ARN 来指定您的 KMS 密钥。

- 要从可用 KMS 密钥列表中进行选择，请选择从您的 AWS KMS 密钥中进行选择，然后从可用密钥列表中选择对称加密 KMS 密钥。确保该 KMS 密钥与您的存储桶位于同一区域。

> **Note**
>
> AWS 托管式密钥 (aws/s3) 和您的客户自主管理型密钥都显示在列表中。但是，使用 S3 清单进行的 SSE-KMS 加密不支持 AWS 托管式密钥 (aws/s3)。

- 要输入 KMS 密钥 ARN，请选择输入 AWS KMS 密钥 ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
- 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

13. 对于其他元数据字段，从下面选择一项或多项以添加到清单报告：

- 大小 - 以字节为单位的对象大小，不包括未完成的分段上传、对象元数据和删除标记的大小。
- Last modified date (上次修改日期) – 对象创建日期或上次修改日期（以较晚者为准）。
- 分段上传 – 指定对象以分段上传形式上传。有关更多信息，请参阅 使用分段上传来上传和复制对象。
- Replication status (复制状态) – 对象的复制状态。有关更多信息，请参阅 获取复制状态信息。
- 加密状态 – 用于加密对象的服务器端加密类型。有关更多信息，请参阅 使用服务器端加密保护数据。
- 桶密钥状态 – 指示是否将 AWS KMS 生成的桶级密钥应用于对象。有关更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本。
- 对象访问控制列表 – 每个对象的访问控制列表（ACL），用于定义哪个 AWS 账户或组被授予对此对象的访问权限以及授予的访问权限类型。有关此字段的更多信息，请参阅使用“对象 ACL”字段。有关 ACL 的更多信息，请参阅 访问控制列表 (ACL) 概述。
- 对象拥有者 – 对象的拥有者。
- 存储类 – 用于存储对象的存储类。
- 智能分层：访问层 – 如果存储在“S3 智能分层”存储类中，则表示对象的访问层（频繁或不频繁）。有关更多信息，请参阅 用于自动优化访问模式不断变化或未知的数据的存储类。
- ETag – 实体标签（ETag）是对象的哈希。ETag 仅反映对于对象的内容的更改，而不反映对于对象的元数据的更改。ETag 可能是该对象数据的 MD5 摘要，也可能不是。是与不是取决于对象的创建方式和加密方式。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 Object。
- 校验和算法 – 表示用于创建对象的校验和的算法。
- 所有对象锁定配置 – 对象的对象锁定状态，包括以下设置：
  - 对象锁定：保留模式 – 应用于对象的保护级别，可以是监管或合规。
  - 对象锁定：保留截止日期 – 在此日期之前无法删除锁定的对象。
  - 对象锁定：依法保留状态 – 锁定对象的依法保留状态。

  有关 S3 对象锁定的更多信息，请参阅S3 对象锁定的工作原理。

有关清单报告内容的更多信息，请参阅 Amazon S3 清单列表。

有关在清单配置中限制对某些可选元数据字段的访问的更多信息，请参阅控制 S3 清单报告配置的创建。

14. 选择创建。

使用 REST API 处理 S3 清单

下面是可用于处理 Amazon S3 清单的 REST 操作。

- DeleteBucketInventoryConfiguration
- GetBucketInventoryConfiguration
- ListBucketInventoryConfigurations
- PutBucketInventoryConfiguration

设置清单完成时的 Amazon S3 事件通知

您可以设置 Amazon S3 事件通知以在创建清单校验和文件后接收通知，该通知将指示库存列表已添加到目标存储桶。清单是目标位置的所有库存列表的最新列表。

Amazon S3 可以将事件发布到 Amazon Simple Notification Service (Amazon SNS) 主题、Amazon Simple Queue Service (Amazon SQS) 队列或 AWS Lambda 函数。有关更多信息，请参阅 Amazon S3 事件通知。

以下通知配置定义了新添加到目标存储桶的所有 `manifest.checksum` 文件将由 AWS Lambda `cloud-function-list-write` 处理。

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>destination-prefix/source-bucket</Value>
              </FilterRule>
              <FilterRule>
```

```
                    <Name>suffix</Name>
                    <Value>checksum</Value>
                </FilterRule>
            </S3Key>
        </Filter>
        <Cloudcode>arn:aws:lambda:us-west-2:222233334444:cloud-function-list-write</
Cloudcode>
        <Event>s3:ObjectCreated:*</Event>
    </QueueConfiguration>
    </NotificationConfiguration>
```

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的AWS Lambda 与 Amazon S3 结合使用。

查找清单列表

在清单列表发布后，Manifest 文件将发布到目标存储桶中的以下位置。

```
destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.json
 destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.checksum
 destination-prefix/source-bucket/config-ID/hive/dt=YYYY-MM-DD-HH-MM/symlink.txt
```

- *destination-prefix* 是对象键名称前缀，可以选择在资源清单配置中指定。您可以使用此前缀对目标存储桶内公共位置中的所有清单列表文件进行分组。
- *source-bucket* 是清单列表所针对的源存储桶。添加源存储桶名称是为了防止在将不同源存储桶中的多个清单报告发送到同一目标存储桶时发生冲突。
- 为防止从同一源存储桶发送到同一目标存储桶的多个清单报告发生冲突，添加了 *config-ID*。*config-ID* 来自清单报告配置，是在设置过程中定义的报告的名称。
- *YYYY-MM-DDTHH-MMZ* 是时间戳，包含清单报告生成过程开始扫描存储桶的开始时间和日期；例如，2016-11-06T21-32Z。
- manifest.json 是 Manifest 文件。
- manifest.checksum 是 manifest.json 文件内容的 MD5 哈希。
- symlink.txt 是 Apache Hive 兼容的清单文件。

清单列表每天或每周发布到目标存储桶中的以下位置。

```
destination-prefix/source-bucket/config-ID/data/example-file-name.csv.gz
...
destination-prefix/source-bucket/config-ID/data/example-file-name-1.csv.gz
```

- *destination-prefix* 是对象键名称前缀，可以选择在资源清单配置中指定。您可以使用此前缀对目标存储桶内公共位置中的所有清单列表文件进行分组。
- *source-bucket* 是清单列表所针对的源存储桶。添加源存储桶名称是为了防止在将不同源存储桶中的多个清单报告发送到同一目标存储桶时发生冲突。
- *example-file-name*.csv.gz 是 CSV 清单文件之一。ORC 清单名称以文件扩展名 .orc 结尾，Parquet 清单名称以文件扩展名 .parquet 结尾。

清单 Manifest

Manifest 文件 manifest.json 和 symlink.txt 描述清单文件的位置。每次交付新的清单列表时，它均带有一组新的 Manifest 文件。这些文件可能会互相覆盖。在启用了版本控制的存储桶中，Amazon S3 会创建清单文件的新版本。

manifest.json 文件中包含的每个 Manifest 均提供了有关清单的元数据和其他基本信息。这些信息包含：

- 源存储桶名称
- 目标存储桶名称
- 清单的版本
- 采用纪元日期格式的创建时间戳，包含清单报告生成过程开始扫描存储桶的开始时间和日期
- 清单文件的格式和架构
- 目标存储桶中清单文件的列表

每当写入 manifest.json 文件后，它都会附带一个 manifest.checksum 文件（作为 manifest.json 文件内容的 MD5 哈希）。

Example **manifest.json** 文件中的清单 Manifest

以下示例显示了采用 CSV、ORC 和 Parquet 格式的清单的 manifest.json 文件中的清单列表。

CSV

下面是 CSV 格式清单的 manifest.json 文件中的 Manifest 示例。

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-inventory-destination-bucket",
```

```
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
 Size, LastModifiedDate, ETag, StorageClass, IsMultipartUploaded,
 ReplicationStatus, EncryptionStatus, ObjectLockRetainUntilDate, ObjectLockMode,
 ObjectLockLegalHoldStatus, IntelligentTieringAccessTier, BucketKeyStatus,
 ChecksumAlgorithm, ObjectAccessControlList, ObjectOwner",
    "files": [
        {
            "key": "Inventory/example-source-bucket/2016-11-06T21-32Z/
files/939c6d46-85a9-4ba8-87bd-9db705a579ce.csv.gz",
            "size": 2147483647,
            "MD5checksum": "f11166069f1990abeb9c97ace9cdfabc"
        }
    ]
}
```

ORC

下面是 ORC 格式清单的 `manifest.json` 文件中的 Manifest 示例。

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "ORC",
    "fileSchema":
 "struct<bucket:string,key:string,version_id:string,is_latest:boolean,is_delete_marker:boole
    "files": [
        {
            "key": "inventory/example-source-bucket/data/
d794c570-95bb-4271-9128-26023c8b4900.orc",
            "size": 56291,
            "MD5checksum": "5925f4e78e1695c2d020b9f6eexample"
        }
    ]
}
```

Parquet

下面是 `manifest.json` 格式清单的 Parquet 文件中的清单示例。

```
{
    "sourceBucket": "example-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "Parquet",
    "fileSchema": "message s3.inventory { required binary bucket (UTF8);
 required binary key (UTF8); optional binary version_id (UTF8); optional boolean
 is_latest; optional boolean is_delete_marker; optional int64 size; optional
 int64 last_modified_date (TIMESTAMP_MILLIS); optional binary e_tag (UTF8);
 optional binary storage_class (UTF8); optional boolean is_multipart_uploaded;
 optional binary replication_status (UTF8); optional binary encryption_status
 (UTF8); optional int64 object_lock_retain_until_date (TIMESTAMP_MILLIS); optional
 binary object_lock_mode (UTF8); optional binary object_lock_legal_hold_status
 (UTF8); optional binary intelligent_tiering_access_tier (UTF8); optional binary
 bucket_key_status (UTF8); optional binary checksum_algorithm (UTF8); optional
 binary object_access_control_list (UTF8); optional binary object_owner (UTF8);}",
    "files": [
        {
           "key": "inventory/example-source-bucket/data/
d754c470-85bb-4255-9218-47023c8b4910.parquet",
            "size": 56291,
            "MD5checksum": "5825f2e18e1695c2d030b9f6eexample"
        }
    ]
}
```

`symlink.txt` 文件是一个与 Apache Hive 兼容的清单文件，它使 Hive 能够自动发现清单文件及其关联的数据文件。与 Hive 兼容的清单文件可以与 Hive 兼容的服务 Athena 和 Amazon Redshift Spectrum 一起使用。它还可用于与 Hive 兼容的应用程序，包括 Presto、Apache Hive、Apache Spark 以及许多其他应用程序。

> **Important**
>
> `symlink.txt` Apache Hive 兼容的清单文件当前不适用于 AWS Glue。
> 对于 ORC 和 Parquet 格式的清单文件，不支持使用 Apache Hive 和 Apache Spark 读取 `symlink.txt` 文件。

使用 Amazon Athena 查询 Amazon S3 清单

您可以在 Athena 可用的所有区域中，通过 Amazon Athena 使用标准 SQL 查询来查询 Amazon S3 清单文件。要检查 AWS 区域 可用性，请参阅 AWS 区域 表。

Athena 可以采用经 Apache 优化的行列式（ORC）、Apache Parquet 或逗号分隔值（CSV）格式来查询 Amazon S3 清单文件。当您使用 Athena 查询清单文件时，我们建议您使用 ORC 格式或 Parquet 格式的清单文件。ORC 和 Parquet 格式提供了更快的查询性能并能够降低查询成本。ORC 和 Parquet 是自我描述、可感知类型的列式文件格式，专为 Apache Hadoop 而设计。列式格式允许读取器仅读取、解压缩并处理当前查询所需的列。所有 AWS 区域均提供 ORC 和 Parquet 格式的 Amazon S3 清单。

使用 Athena 查询 Amazon S3 清单文件

1. 创建 Athena 表。有关创建表的信息，请参阅《Amazon Athena 用户指南》中的在 Amazon Athena 中创建表。
2. 使用以下示例查询模板之一创建查询，具体取决于您查询的是 ORC 格式、Parquet 格式还是 CSV 格式的清单报告。

   - 当您使用 Athena 查询 ORC 格式的清单报告时，请使用以下示例查询作为模板。

     下面的示例查询在 ORC 格式的清单报告中包含所有可选字段。

     要使用此示例查询，请执行以下操作：

     - 将 *your_table_name* 替换为您创建的 Athena 表的名称。
     - 删除您没有为清单选择的所有可选字段，以便查询对应于您的清单的选定字段。
     - 根据您的配置替换以下桶名称和清单位置（配置 ID）。

       s3://*amzn-s3-demo-bucket*/*config-ID*/hive/

     - 将 `projection.dt.range` 下的 *2022-01-01-00-00* 日期替换为在 Athena 中对数据进行分区的时间范围的第一天。有关更多信息，请参阅 Athena 中的分区数据。

```
CREATE EXTERNAL TABLE your_table_name(
         bucket string,
         key string,
         version_id string,
         is_latest boolean,
         is_delete_marker boolean,
         size bigint,
```

```
         last_modified_date timestamp,
         e_tag string,
         storage_class string,
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date bigint,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

- 当您使用 Athena 查询 Parquet 格式的清单报告时，请使用 ORC 格式报告的示例查询。但是，请使用以下 Parquet SerDe 代替 `ROW FORMAT SERDE` 语句中的 ORC SerDe。

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
```

- 当您使用 Athena 查询 CSV 格式的清单报告时，请使用以下示例查询作为模板。

  下面的示例查询在 CSV 格式的清单报告中包含所有可选字段。

  要使用此示例查询，请执行以下操作：

  - 将 *`your_table_name`* 替换为您创建的 Athena 表的名称。
  - 删除您没有为清单选择的所有可选字段，以便查询对应于您的清单的选定字段。

- 根据您的配置替换以下桶名称和清单位置（配置 ID）。

  s3://*amzn-s3-demo-bucket*/*config-ID*/hive/

- 将 projection.dt.range 下的 *2022-01-01-00-00* 日期替换为在 Athena 中对数据进行分区的时间范围的第一天。有关更多信息，请参阅 Athena 中的分区数据。

```
CREATE EXTERNAL TABLE your_table_name(
         bucket string,
         key string,
         version_id string,
         is_latest boolean,
         is_delete_marker boolean,
         size string,
         last_modified_date string,
         e_tag string,
         storage_class string,
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date string,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://source-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

3. 现在，您可以对清单运行各种查询，如以下示例所示。将每个 *user input placeholder* 替换为您自己的信息。

```
# Get a list of the latest inventory report dates available.
SELECT DISTINCT dt FROM your_table_name ORDER BY 1 DESC limit 10;

# Get the encryption status for a provided report date.
SELECT encryption_status, count(*) FROM your_table_name WHERE dt = 'YYYY-MM-DD-HH-
MM' GROUP BY encryption_status;

# Get the encryption status for inventory report dates in the provided range.
SELECT dt, encryption_status, count(*) FROM your_table_name
WHERE dt > 'YYYY-MM-DD-HH-MM' AND dt < 'YYYY-MM-DD-HH-MM' GROUP BY dt,
 encryption_status;
```

当您配置 S3 清单以将对象访问控制列表（对象 ACL）字段添加到清单报告时，报告将“对象 ACL”字段的值显示为以 base64 编码的字符串。要以 JSON 格式获取“对象 ACL”字段的解码值，可以使用 Athena 查询此字段。请参阅以下查询示例。有关“对象 ACL”字段的更多信息，请参阅使用“对象 ACL”字段。

```
# Get the S3 keys that have Object ACL grants with public access.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'uri') = 'http://acs.amazonaws.com/groups/global/AllUsers'
```

```
# Get the S3 keys that have Object ACL grantees in addition to the object owner.
WITH grants AS
    (SELECT key,
    from_utf8(from_base64(object_access_control_list)) AS
 object_access_control_list,
```

```
         object_owner,
         CAST(json_extract(from_utf8(from_base64(object_access_control_list)),
         '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))) AS grants_array
    FROM your_table_name)
SELECT key,
       grant,
       objectowner
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE cardinality(grants_array) > 1 AND element_at(grant, 'canonicalId') !=
 object_owner;
```

```
# Get the S3 keys with READ permission that is granted in the Object ACL.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'permission') = 'READ';
```

```
# Get the S3 keys that have Object ACL grants to a specific canonical user ID.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'canonicalId') = 'user-canonical-id';
```

```
# Get the number of grantees on the Object ACL.
SELECT key,
       object_access_control_list,
       json_array_length(json_extract(object_access_control_list,'$.grants')) AS
 grants_count
FROM your_table_name;
```

有关使用 Athena 的更多信息，请参阅 Amazon Athena 用户指南。

将 Amazon S3 库存报告中的空版本 ID 字符串转换为空字符串

> **Note**
>
> 下面的过程只适用于包含所有版本的 Amazon S3 库存报告，并且只有当"所有版本"的报告作为 S3 分批操作在已启用 S3 版本的存储桶上的显示时才适用。您无需为仅指定当前版本的 S3 清单报告转换字符串。

您可以使用 S3 清单报告作为 S3 分批操作的清单。但是，在存储桶上启用 S3 版本控制时，包括所有版本的 S3 清单报告将在版本 ID 字段中用空字符串标记任何空版本控制的对象。当库存报告包含所有对象版本 ID 时，分批操作将 `null` 字符串识别为版本 ID，但不是空字符串。

当 S3 分批操作任务使用“所有版本”S3 清单报告作为清单时，它会使版本 ID 字段中有空字符串的对象上的所有任务失败。要将 S3 库存报告版本 ID 字段中的空字符串转换为分批操作 `null` 字符串，使用以下过程。

更新 Amazon S3 库存报告以便与分批操作一起使用

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 导航至 S3 清单报告。库存报告位于您在配置库存报告时指定的目标存储桶中。有关查找清单报告的更多信息，请参阅 查找清单列表。
   a. 选择目标存储桶。
   b. 选择 文件夹。该文件夹以原始源存储桶命名。

c. 选择以清单配置命名的文件夹。

d. 选择名为 hive 的文件夹旁边的勾选框。在页面顶部，选择 Copy S3 URI（复制 S3 URI）以为文件夹复制 S3 URI。

3. 从 https://console.aws.amazon.com/athena/ 打开 Amazon Athena 控制台。

4. 在查询编辑器中，选择 Settings（设置），然后选择 Manage（管理）。在存储库的 Manage settings（管理设置）页面中，针对 Location of query result（查询结果的位置），选择用于存储查询结果的 S3 存储桶。

5. 在查询编辑器中，使用以下命令创建 Athena 表以保存库存报告中的数据。将 *table_name* 替换为您所选择的名称，然后在 LOCATION 子句中，插入之前复制的 S3 URI。然后选择 Run（运行），以运行查询。

```
CREATE EXTERNAL TABLE table_name(bucket string, key string,
 version_id string) PARTITIONED BY (dt string)ROW FORMAT SERDE
 'org.apache.hadoop.hive.serde2.OpenCSVSerde'STORED AS INPUTFORMAT
 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat' OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat' LOCATION 'Copied S3 URI';
```

6. 要清除查询编辑器，请选择 Clear（清除）。然后使用以下命令将清单报告加载到表中。将 *table_name* 替换为您在上一步所选择的项。然后选择 Run（运行），以运行查询。

```
MSCK REPAIR TABLE table_name;
```

7. 要清除查询编辑器，请选择 Clear（清除）。运行以下 SELECT 查询以检索原始库存报告中的所有条目，并将任何空版本 ID 替换为 null 字符串。将 *table_name* 替换为先前所选择的项，然后将 WHERE 子句中的 *YYYY-MM-DD-HH-MM* 替换为您想要此工具运行的库存报告的日期。然后选择 Run（运行），以运行查询。

```
SELECT bucket as Bucket, key as Key, CASE WHEN version_id = '' THEN 'null' ELSE
 version_id END as VersionId FROM table_name WHERE dt = 'YYYY-MM-DD-HH-MM';
```

8. 返回 Amazon S3 控制台 (https://console.aws.amazon.com/s3/)，然后导航到稍早您为 Location of query result（查询结果位置）选择的 S3 存储桶。其内部应该有一系列以日期结束的文件夹。

   例如，您将看到类似于 s3://***DOC-EXAMPLE-BUCKET***/***query-result-location***/Unsaved/2021/10/07/ 的内容。您将看到包含您运行的 SELECT 查询的结果的 .csv 文件。

   选择具有最新修改日期的 CSV 文件。将此文件下载到本地计算机以进行下一步。

9. 生成的 CSV 文件包含标题行。要使用此 CSV 文件作为 S3 分批操作任务的输入，您必须移除标题行，因为分批操作不支持 CSV 清单上的标题行。

   要移除标题行，可以对文件运行以下命令之一。将 *file.csv* 替换为 CSV 文件的名称。

   对于 macOS 和 Linux 计算机，在终端窗口运行 `tail` 命令。

   ```
   tail -n +2 file.csv > tmp.csv && mv tmp.csv file.csv
   ```

   对于 Windows 计算机，在 Windows PowerShell 窗口中运行以下脚本。将 *File-location* 替换为文件的路径，将 *file.csv* 替换为文件名。

   ```
   $ins = New-Object System.IO.StreamReader File-location\file.csv
   $outs = New-Object System.IO.StreamWriter File-location\temp.csv
   try {
       $skip = 0
       while ( !$ins.EndOfStream ) {
           $line = $ins.ReadLine();
           if ( $skip -ne 0 ) {
               $outs.WriteLine($line);
           } else {
               $skip = 1
           }
       }
   } finally {
       $outs.Close();
       $ins.Close();
   }
   Move-Item File-location\temp.csv File-location\file.csv -Force
   ```

10. 从 CSV 文件中移除标题行后，您可以在 S3 分批操作任务中将其用作清单。将 CSV 文件上载到 S3 存储桶或您选择的位置，然后使用 CSV 文件作为清单创建分批操作任务。

    有关创建 S3 分批操作任务的更多信息，请参阅 创建 S3 批量操作任务。

使用“对象 ACL”字段

Amazon S3 清单报告包含 S3 源桶中对象的列表以及每个对象的元数据。“对象访问控制列表（ACL）”字段是 Amazon S3 清单中提供的一个元数据字段。具体而言，“对象 ACL”字段包含每个对象

的访问控制列表（ACL）。对象的 ACL 定义了哪个 AWS 账户或组被授予对此对象的访问权限以及所授予的访问权限类型。有关更多信息，请参阅 访问控制列表 (ACL) 概述和 Amazon S3 清单列表。

Amazon S3 清单报告中的“对象 ACL”字段以 JSON 格式定义。JSON 数据包含以下字段：

- `version` – 清单报告中“对象 ACL”字段格式的版本。它采用日期格式 `yyyy-mm-dd`。
- `status` – 可能的值为 `AVAILABLE` 或 `UNAVAILABLE`，指明对象 ACL 是否可用于某个对象。当对象 ACL 的状态为 `UNAVAILABLE` 时，清单报告中“对象拥有者”字段的值也是 `UNAVAILABLE`。
- `grants` – 被授权者/权限对，列出了对象 ACL 授予的每个被授权者的权限状态。被授权者的可用值为 `CanonicalUser` 和 `Group`。有关被授权者的更多信息，请参阅访问控制列表中的被授权者。

  对于类型为 `Group` 的被授权者，被授权者/权限对包括以下属性：

  - `uri` – 预定义的 Amazon S3 组。
  - `permission` – 所授予的对于对象的 ACL 权限。有关更多信息，请参阅对于对象的 ACL 权限。
  - `type` – 类型为 `Group`，表示被授权者是组。

  对于类型为 `CanonicalUser` 的被授权者，被授权者/权限对包括以下属性：

  - `canonicalId`— 一种经过混淆处理的 AWS 账户 ID 形式。AWS 账户的规范用户 ID 特定于该账户。您可以检索规范用户 ID。有关更多信息，请参阅《AWS 账户管理参考指南》中的 Find the canonical user ID for your AWS 账户。

    > **Note**
    >
    > 如果 ACL 中的被授权者是 AWS 账户的电子邮件地址，S3 清单使用该 AWS 账户的 `canonicalId` 和 `CanonicalUser` 类型来指定此授权者。有关更多信息，请参阅访问控制列表中的被授权者。

  - `permission` – 所授予的对于对象的 ACL 权限。有关更多信息，请参阅对于对象的 ACL 权限。
  - `type` – 类型为 `CanonicalUser`，表示被授权者是 AWS 账户。

以下示例显示了“对象 ACL”字段的可能值，采用 JSON 格式：

```
{
    "version": "2022-11-10",
    "status": "AVAILABLE",
    "grants": [{
        "uri": "http://acs.amazonaws.com/groups/global/AllUsers",
        "permission": "READ",
```

```
        "type": "Group"
    }, {
        "canonicalId": "example-canonical-id",
        "permission": "FULL_CONTROL",
        "type": "CanonicalUser"
    }]
}
```

Note

“对象 ACL”字段以 JSON 格式定义。清单报告将“对象 ACL”字段的值显示为以 base64 编码的字符串。
例如，假设您有以下采用 JSON 格式的“对象 ACL”字段：

```
{
        "version": "2022-11-10",
        "status": "AVAILABLE",
        "grants": [{
            "canonicalId": "example-canonical-user-ID",
            "type": "CanonicalUser",
            "permission": "READ"
        }]
}
```

“对象 ACL”字段经过编码并显示为以下以 base64 编码的字符串：

```
eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2FsSW
```

要以 JSON 格式获取“对象 ACL”字段的解码值，可以在 Amazon Athena 中查询此字段。有关示例查询，请参阅使用 Amazon Athena 查询 Amazon S3 清单。

复制对象概述

您可以使用复制来启用跨 Amazon S3 存储桶自动以异步方式复制对象。为对象复制配置的存储桶可由相同 AWS 账户 或不同账户拥有。您可以将对象复制到单个目标存储桶或多个目标存储桶。目标存储桶可以位于不同的 AWS 区域，也可以与源存储桶位于同一区域内。

复制有两种类型：实时复制 和按需复制。

- 实时复制：想要在向源存储桶写入新的和更新的对象的同时，自动复制这些对象，请使用实时复制。实时复制不会复制在设置复制之前就存在于存储桶中的任何对象。想要复制在设置复制之前就存在的对象，请使用按需复制。
- 按需复制：想要按需从源存储桶中复制现有对象到一个或多个目标存储桶，请使用 S3 批量复制。有关复制现有对象的更多信息，请参阅 何时使用 S3 分批复制。

实时复制有两种形式：跨区域复制（CRR）和单区域复制（SRR）。

- 跨区域复制（CRR）：可以使用 CRR 跨不同 AWS 区域中的 S3 存储桶复制对象。有关 CRR 的更多信息，请参阅the section called “何时使用跨区域复制”。
- 单区域复制（SRR）：可以使用 SRR 跨同一 AWS 区域中的 Amazon S3 存储桶复制对象。有关 SRR 的更多信息，请参阅the section called “何时使用同区域复制”。

主题

为什么使用复制？

复制可帮助您实现以下功能：

- 复制对象，同时保留元数据 – 您可以使用复制来制作对象的副本，并保留所有元数据（例如原始对象创建时间和版本 ID）。如果您必须确保副本与源对象完全相同，则此功能非常重要。

- 将对象复制到不同的存储类 – 您可以使用复制直接将对象置于目标存储桶中的 S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive 或其他存储类中。您还可以将数据复制到相同的存储类，并在目标存储桶上使用生命周期配置，以在对象老化时将其移动到较冷的存储类。
- 在不同的所有权下维护对象副本 – 无论谁拥有源对象，您都可以指示 Amazon S3 将副本的所有权更改为拥有目标存储桶的 AWS 账户。这称为拥有者覆盖 选项。您可以使用此选项来要限制对对象副本的访问权限。
- 将对象存储在多个 AWS 区域 – 要确保数据保存位置的地理差异，您可以在不同的 AWS 区域 中设置多个目标存储桶。此功能可能会帮助您满足某些合规性要求。
- 在 15 分钟内复制对象 – 要在可预测的时间范围内在同一 AWS 区域 或跨不同区域复制您的数据，您可以使用 S3 Replication Time Control（S3 RTC）。S3 RTC 在 15 分钟内复制 Amazon S3 中存储的 99.99% 的新对象（由服务等级协议提供支持）。有关更多信息，请参阅 the section called “使用 S3 Replication Time Control”。

  > **Note**
  >
  > S3 RTC 不适用于分批复制。分批复制是一项按需复制任务，可以使用 S3 批量操作进行跟踪。有关更多信息，请参阅 跟踪任务状态和完成报告。

- 同步存储桶、复制现有对象以及之前无法复制或已复制的对象 – 要同步存储桶和复制现有对象，请使用分批复制作为按需复制操作。有关何时使用分批复制的更多信息，请参阅 何时使用 S3 分批复制。
- 复制对象并失效转移到另一个 AWS 区域中的存储桶 - 要在数据复制期间使所有元数据和对象在存储桶之间保持同步，请在配置 Amazon S3 多区域接入点失效转移控制之前使用双向复制规则。双向复制规则有助于确保在将数据写入流量失效转移到的 S3 存储桶时，该数据随后被复制回源存储桶。

何时使用跨区域复制

S3 跨区域复制（CRR）用于在不同 AWS 区域中跨 Amazon S3 存储桶复制对象。CRR 可帮助您：

- 满足合规性要求 – 虽然 Amazon S3 默认跨多个地理位置较远的可用区存储数据，但是合规性要求所规定的数据存储距离可能更远。为满足这些要求，可以使用跨区域复制在远距离 AWS 区域 之间复制数据。
- 最大限度减少延迟 – 如果客户处于两个地理位置，您可以在地理位置与用户较近的AWS 区域中维护对象副本，从而最大限度缩短访问对象时的延迟。
- 提高操作效率 – 如果您在两个不同 AWS 区域 中具有分析同一组对象的计算集群，则可以选择在这些区域中维护对象副本。

何时使用同区域复制

同区域复制（SRR）用于在同一 AWS 区域中跨 Amazon S3 存储桶复制对象。SRR 可帮助您：

- 将日志聚合到单个存储桶 – 如果您在多个存储桶中或者跨多个账户存储日志，则可以轻松地将日志复制到区域中的单个存储桶。这样做实现了在单个位置更简单地处理日志。
- 在生产账户和测试账户之间配置实时复制 – 如果您或您的客户有使用相同数据的生产账户和测试账户，您可以在那些账户之间复制对象，同时保留对象元数据。
- 遵守数据主权法律 – 您可能需要将数据的多个副本存储在特定区域内的单独 AWS 账户 中。当合规性法规不允许数据离开您的国家/地区时，同区域复制可帮助您自动复制关键数据。

何时使用双向复制

- 跨多个 AWS 区域构建共享数据集 – 通过副本修改同步功能，您可以轻松地对于复制对象复制元数据更改，例如对象访问控制列表（ACL)、对象标签或对象锁定。如果要使所有对象和对象元数据更改保持同步，这种双向复制非常重要。在相同或不同 AWS 区域中的两个或更多存储桶之间执行双向复制时，您可以对新的或现有的复制规则启用副本修改同步。
- 在失效转移期间跨区域保持数据同步 – 您可以直接从多区域接入点通过 S3 跨区域复制（CRR）配置双向复制规则，在 AWS 区域中的存储桶之间同步数据。要就何时启动失效转移做出明智的决定，您还可以启用 S3 复制指标，以便监控 Amazon CloudWatch、S3 Replication Time Control（S3 RTC）或多区域接入点中的复制。
- 使应用程序具有高可用性 - 即使在区域流量中断的情况下，您也可以使用双向复制规则在数据复制期间跨存储桶保持所有元数据和对象同步。

何时使用 S3 分批复制

作为按需选项，分批复制将现有对象复制到不同的存储桶。与实时复制不同，这些任务可以根据需要运行。分批复制可帮助您实现以下功能：

- 复制现有对象 – 您可以使用分批复制来复制在配置同区域复制或跨区域复制之前添加到存储桶中的对象。
- 复制之前无法复制的对象 – 您可以筛选分批复制任务以尝试复制状态为 FAILED 的对象。
- 复制已经复制的对象 – 您可能需要将数据的多个副本存储在单独的 AWS 账户 或 AWS 区域 中。分批复制可以将现有对象复制到新添加的目标。

- 复制根据复制规则创建的对象的副本 – 复制配置在目标存储桶中创建对象的副本。对象的副本只能通过分批复制进行复制。

工作负载要求和实时复制

根据您的工作负载要求，某些类型的复制将比其它类型的复制更适合您的应用场景。使用下表来确定针对您的场景应该使用哪种类型的复制，以及是否需要对您的工作负载使用 S3 Replication Time Control（S3 RTC）。S3 RTC 在 15 分钟内复制 Amazon S3 中存储的 99.99% 的新对象 [由服务等级协议（SLA）提供支持]。有关更多信息，请参阅 the section called “使用 S3 Replication Time Control”。

复制工作负载要求比较

工作负载要求	S3 RTC（15 分钟 SLA）	跨区域复制 (CRR)	单区域复制（SRR）
在不同 AWS 账户之间复制对象	是	是	是
在 24-48 小时内，在同一 AWS 区域中复制对象（不受 SLA 支持）	否	否	是
在 24-48 小时内，在不同 AWS 区域之间复制对象（不受 SLA 支持）	否	是	不支持
可预测的复制时间：由 SLA 提供支持，可在 15 分钟内复制 99.9% 的对象	是	否	否

Amazon S3 复制什么内容？

Amazon S3 仅复制为复制配置的存储桶中的特定项目。

主题

使用复制配置会复制什么？

默认情况下，Amazon S3 会复制以下内容：

- 添加复制配置之后创建的对象。
- 未加密的对象。
- 使用客户提供密钥（SSE-C）加密的对象，使用 Amazon S3 托管密钥（SSE-S3）或存储在 AWS Key Management Service 中的 KMS 密钥（SSE-KMS）静态加密的对象。有关更多信息，请参阅 the section called “复制加密对象”。
- 从源对象到副本的对象元数据。有关将元数据从副本复制到源对象的信息，请参阅 使用 Amazon S3 副本修改同步复制元数据更改。
- 仅限存储桶拥有者有权读取其对象和访问控制列表（ACL）的源存储存储桶中的对象。

  有关资源所有权的更多信息，请参阅 Amazon S3 存储桶和对象所有权。
- 对象 ACL 更新（但源存储桶和目标存储桶不是由相同账户拥有时指示 Amazon S3 更改副本所有权的情况除外）。

  有关更多信息，请参阅 更改副本拥有者。

  在 Amazon S3 能够使两个 ACL 同步之前，可能需要花一些时间。这一所有权变更仅适用于在向存储桶添加复制配置之后创建的对象。
- 对象标签（如果有）。
- S3 对象锁定保留信息（如果有）。

  如果 Amazon S3 复制应用了保留信息的对象，它会将相同的保留控制功能应用于您的副本，从而替换在目标存储桶上配置的默认保留期。如果未对您源存储桶中的对象应用保留控制功能，并且您复制到设置了默认保留期的目标存储桶，则目标存储桶的默认保留期将应用于对象副本。有关更多信息，请参阅 使用 S3 对象锁定。

删除操作对复制操作有何影响

如果您从源存储桶中删除对象，则默认情况下会执行以下操作：

- 如果您发出 DELETE 请求而未指定对象版本 ID，Amazon S3 会添加一个删除标记。Amazon S3 将按如下所示处理该删除标记：
  - 如果您使用的是最新版本的复制配置（即在复制配置规则中指定 `Filter` 元素），Amazon S3 默认不会复制该删除标记。但是，您可以将删除标记复制添加到非基于标记的规则。有关更多信息，请参阅 在存储桶之间复制删除标记。
  - 如果您未指定 `Filter` 元素，则 Amazon S3 将假定复制配置为版本 V1，并复制用户操作产生的删除标记。但是，如果 Amazon S3 因生命周期操作删除了对象，则删除标记不会复制到目标存储桶。
- 如果您在 `DELETE` 请求中指定一个要删除的对象版本 ID，Amazon S3 会在源存储桶中删除该对象版本。但不会将删除操作复制到目标存储桶中。换句话说，它不会从目标存储桶中删除同一对象版本。这会防止恶意删除数据。

使用复制配置不会复制什么？

默认情况下，Amazon S3 不复制以下内容：

- 源存储桶中作为另一个复制规则所建副本的对象。例如，假设您配置的复制中，存储桶 A 是源，存储桶 B 是目标。现在假设您添加另一个复制配置，其中存储桶 B 是源，而存储桶 C 是目标。在这种情况下，存储桶 B 中作为存储桶 A 中对象的副本的对象不会复制到存储桶 C。

  要复制属于副本的对象，请使用分批复制。请通过 复制现有对象 了解有关配置分批复制的更多信息。
- 源存储桶中已复制到其他目标的对象。例如，如果您在现有复制配置中更改目标存储桶，则 Amazon S3 不会再次复制对象。

  要复制之前已复制的对象，请使用分批复制。请通过 复制现有对象 了解有关配置分批复制的更多信息。
- 批处理复制不支持重新复制从目标存储桶中使用对象的版本 ID 删除的对象。要重新复制这些对象，您可以使用分批复制任务将源对象复制到位。将这些对象复制到位会在源存储桶中创建对象的新版本，并自动启动到目标的复制。有关如何使用分批复制的更多信息，请参阅使用分批操作复制对象的示例。
- 默认情况下，从不同的 AWS 账户 进行复制时，添加到源存储桶的删除标记不会被复制。

有关如何复制删除标记的更多信息，请参阅 在存储桶之间复制删除标记。

- 存储在 S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive、S3 Intelligent-Tiering Archive Access 或 S3 Intelligent-Tiering Deep Archive Access 存储类或层中的对象。在还原这些对象并将它们复制到不同的存储类之前，您无法复制它们。

  要了解有关 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 的更多信息，请参阅极少访问的对象的存储类。

  有关 S3 Intelligent-Tiering 的更多信息，请参阅Amazon S3 Intelligent-Tiering。

- 源存储存储桶中存储桶拥有者没有足够权限进行复制的对象。

  有关对象拥有者如何向存储桶拥有者授予权限的信息，请参阅在授予上传对象的跨账户权限的同时，确保存储桶拥有者拥有完全控制权。

- 对存储桶级别子资源进行的更新。

  例如，如果您更改生命周期配置或向源存储桶添加通知配置，则这些更改不会应用于目标存储桶。此功能使您可以在源存储桶和目标存储桶中具有不同的配置。

- 由生命周期配置执行的操作。

  例如，如果仅对源存储桶启用了生命周期配置，则 Amazon S3 会为过期对象创建删除标记，但不会复制这些标记。如果您希望对源存储桶和目标存储桶应用相同的生命周期配置，请对这两个存储桶启用相同的生命周期配置。有关生命周期配置的更多信息，请参阅 管理存储生命周期。

- 当您在实时复制中使用基于标签的复制规则时，必须在 `PutObject` 操作中使用匹配的复制规则标签来标记新对象。否则，将不会复制对象。如果在 `PutObject` 操作后标记了对象，也不会复制这些对象。

  要复制在 `PutObject` 操作之后标记的对象，您必须使用 S3 批量复制。有关分批复制的更多信息，请参阅 复制现有对象。

默认存储桶加密如何影响复制

在为复制目标存储桶启用默认加密后，将应用以下加密行为：

- 如果未对源存储桶中的对象进行加密，则将使用目标存储桶的默认加密设置对目标存储桶中的副本对象进行加密。因此，源对象的实体标签（ETag）与副本对象的 ETag 不同。如果您有使用 ETag 的应用程序，则必须更新这些应用程序以弥补这种差异。

- 如果使用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）、具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）或具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）来加密源存储桶中的对象，则目标存储桶中的副本对象使用与源对象相同类型的加密。不会使用目标存储桶的默认加密设置。

复制的要求和注意事项

Amazon S3 复制具有以下要求：

- 源存储桶拥有者必须已为其账户启用源和目标 AWS 区域。目标存储桶拥有者必须已为其账户启用目标区域。

  有关启用和禁用 AWS 区域的更多信息，请参阅《AWS 一般参考》中的管理 AWS 区域。
- 源存储桶和目标存储桶必须均已启用版本控制。有关版本控制的更多信息，请参阅 在 S3 存储桶中使用版本控制。
- Amazon S3 必须有权代表您将对象从源存储桶复制到目标存储桶。有关这些权限的更多信息，请参阅 为实时复制设置权限。
- 如果源存储桶的拥有者不拥有该存储桶中的对象，则对象拥有者必须使用对象访问控制列表（ACL）对该存储桶拥有者授予 READ 和 READ_ACP 权限。有关更多信息，请参阅 访问控制列表 (ACL) 概述。
- 如果源存储桶已启用 S3 对象锁定，目标存储桶也必须启用 S3 对象锁定。

  要在启用了对象锁定的存储桶上启用复制，您必须使用 AWS Command Line Interface、REST API 或 AWS SDK。有关更多一般信息，请参阅 使用 S3 对象锁定。

  > **Note**
  >
  > 您必须在用于设置复制的 AWS Identity and Access Management (IAM) 角色中授予对源 S3 存储桶的两项新权限。这两项新权限是 `s3:GetObjectRetention` 和 `s3:GetObjectLegalHold`。如果这个角色有 `s3:Get*` 权限，则它已经满足要求了。有关更多信息，请参阅 为实时复制设置权限。

有关更多信息，请参阅 设置实时复制。

如果您要在跨账户方案（其中，源存储桶和目标存储桶由不同的 AWS 账户拥有）中设置复制配置，则需要满足以下附加要求：

- 目标存储桶的拥有者必须使用存储桶策略向源存储桶的拥有者授予复制对象的权限。有关更多信息，请参阅 在源存储桶和目标存储桶由不同的 AWS 账户 拥有时授予权限。
- 目标存储桶不能配置为申请方付款存储桶。有关更多信息，请参阅 使用申请方付款存储桶进行存储传输和使用。

复制的注意事项

在创建复制配置之前，请注意以下注意事项。

主题

- 生命周期配置和对象副本
- 版本控制配置和复制配置
- 将 S3 复制与 S3 Intelligent-Tiering 结合使用
- 日志记录配置和复制配置
- CRR 和目标区域
- S3 批量复制
- S3 Replication Time Control

生命周期配置和对象副本

Amazon S3 复制对象所需的时间取决于对象大小。对于大型对象，可能需要几个小时。虽然可能需要一段时间之后副本才能出现在目标存储桶中，但创建副本所需的时间与创建源存储桶中的对应对象所需的时间相同。如果您对目标存储桶启用了生命周期配置，生命周期规则遵循对象的原始创建时间，而不是副本在目标存储桶中可用的时间。

复制配置需要启用版本控制的存储桶。在存储桶上启用版本控制时，请注意以下几点：

- 如果您具有对象过期生命周期配置，在启用版本控制后，请添加 `NonCurrentVersionExpiration` 策略以保持与启用版本控制之前相同的永久删除行为。
- 如果具有转换生命周期配置，则在启用版本控制后，考虑添加 `NonCurrentVersionTransition` 策略。

版本控制配置和复制配置

对存储桶配置复制时，源存储桶和目标存储桶都必须启用版本控制。对源存储桶和目标存储桶启用版本控制并对源存储桶配置复制后，您会遇到以下问题：

- 如果您尝试对源存储桶禁用版本控制，则 Amazon S3 会返回错误。您必须先删除复制配置，然后才能对源存储桶禁用版本控制。
- 如果您对目标存储桶禁用版本控制，则复制会失败。源对象的复制状态为 FAILED。

将 S3 复制与 S3 Intelligent-Tiering 结合使用

S3 Intelligent-Tiering 是一种存储类，旨在通过自动将数据移动到最具成本效益的访问层来优化存储成本。每月只需支付少量的对象监控和自动化费用，S3 Intelligent-Tiering 即可监控访问模式并将未访问的对象移动到成本较低的访问层。

使用 S3 批量复制来复制存储在 S3 Intelligent-Tiering 中的对象或调用 CopyObject 或 UploadPartCopy 将构成访问。在这些情况下，复制操作的源对象是分层的。

有关 S3 Intelligent-Tiering 的更多信息，请参阅Amazon S3 Intelligent-Tiering。

日志记录配置和复制配置

如果 Amazon S3 将日志传送到已启用复制的存储桶，它将复制日志对象。

如果对源存储桶或目标存储桶启用了服务器访问日志（使用服务器访问日志记录来记录请求）或 AWS CloudTrail 日志（使用 AWS CloudTrail 记录 Amazon S3 API 调用），Amazon S3 将在这些日志中包含复制相关请求。例如，Amazon S3 将记录它复制的每个对象。

CRR 和目标区域

Amazon S3 跨区域复制（CRR）用于在不同 AWS 区域 中跨 S3 存储桶复制对象。根据您的业务需求或成本考虑，您可能选择了区域作为目标存储桶。例如，区域间数据传输费用因您所选的区域而异。

例如，假设您选择了美国东部（弗吉尼亚州北部）（us-east-1）作为您的源存储桶区域。如果您选择了美国西部（俄勒冈州）（us-west-2）作为目标存储桶区域，则需要支付超出选择美国东部（俄亥俄州）（us-east-2）区域时的费用。有关定价信息，请参阅 Amazon S3 定价中的“数据传输定价”。

同区域复制 (SRR) 不会产生相关的数据传输费用。

S3 批量复制

有关批量复制的注意事项的信息，请参阅 S3 分批复制注意事项。

S3 Replication Time Control

有关 S3 Replication Time Control（S3 RTC）的最佳实践和注意事项的信息，请参阅 S3 RTC 的最佳实践和准则。

设置实时复制

> **Note**
>
> 在您设置复制之前就存在的对象将不会自动复制。换句话说，Amazon S3 不以回溯方式复制对象。要复制在复制配置之前创建的对象，请使用 S3 分批复制。请通过 复制现有对象 了解有关配置分批复制的更多信息。

要启用实时复制，即同区域复制（SRR）或跨区域复制（CRR），请向源存储桶添加复制配置。该配置会指示 Amazon S3 按照指定的方式复制对象。在复制配置中，您必须提供以下内容：

- 目标存储桶 – 您希望 Amazon S3 将对象复制到的存储桶。
- 要复制的对象 – 您可以复制源存储桶中的所有对象或对象子集。通过在配置中提供一个键名前缀和/或一个或多个对象标签，可标识子集。

  例如，如果您配置复制规则，仅复制键名前缀为 `Tax/` 的对象，则 Amazon S3 仅复制键为 `Tax/doc1` 或 `Tax/doc2` 之类的对象。但它不复制具有键 `Legal/doc3` 的对象。如果您同时指定前缀和一个或多个标签，则 Amazon S3 仅复制具有特定键前缀和这些标签的对象。
- AWS Identity and Access Management（IAM）角色 – Amazon S3 代入此 IAM 角色以代表您复制对象。

除了这些最低要求，您还可以选择以下选项：

- 副本存储类 – 默认情况下，Amazon S3 使用与源对象相同的存储类来存储对象副本。您可以为副本指定其他存储类。
- 副本所有权 – Amazon S3 假定对象副本继续由源对象的拥有者拥有。因此，在复制对象时，它还会复制对应的对象访问控制列表 (ACL) 或 S3 对象拥有权设置。如果源存储桶和目标存储桶由不同的 AWS 账户拥有，您可以配置复制以将副本的拥有者更改为拥有目标存储桶的 AWS 账户。

您可以使用 REST API、AWS SDK、AWS Command Line Interface (AWS CLI) 或 Amazon S3 控制台配置复制。

Amazon S3 还提供了 API 操作来于支持设置复制规则。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的以下主题：

- PutBucketReplication

- GetBucketReplication
- DeleteBucketReplication

主题

- 复制配置
- 为实时复制设置权限
- 配置实时复制的示例

复制配置

Amazon S3 以 XML 形式存储复制配置。在复制配置 XML 文件中，您将指定一个 AWS Identity and Access Management (IAM) 角色以及一个或多个规则。

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

Amazon S3 无法在未经您的权限的情况下复制对象。您将向复制配置中指定的 IAM 角色授予权限。Amazon S3 将担任该 IAM 角色以代表您复制对象。您必须先向 IAM 角色授予必要的权限。有关管理权限的更多信息，请参阅 为实时复制设置权限。

在以下场景的复制配置中添加一个规则：

- 您希望复制所有对象。
- 您希望复制对象子集。通过在规则中添加一个筛选条件，可标识对象子级。在该筛选条件中，指定对象键前缀、标签或二者的组合以标识要向其应用规则的对象子集。筛选条件指向与您指定的确切值相匹配的对象。

如果要复制其他对象子集，请在复制配置中添加多个规则。在每个规则中，指定一个选择不同对象子集的筛选条件。例如，您可能选择了键前缀为 `tax/` 或 `document/` 的复制对象。要做到这一点，您需要

添加两个规则，一个规则指定 tax/ 键前缀筛选条件，另一个指定 document/ 键前缀。有关对象键前缀的更多信息，请参阅使用前缀组织对象。

以下各节提供了更多信息。

主题

- 基本规则配置
- 可选：指定筛选条件
- 其他目标配置
- 复制配置示例
- 向后兼容性

基本规则配置

每个规则必须包含规则的状态和优先级。规则还必须指示是否复制删除标记。

- Status 通过使用值 Enabled 或 Disabled 指示规则是启用还是禁用。如果规则处于禁用状态，则 Amazon S3 不会执行规则中指定的操作。
- Priority 指示当两个或更多复制规则冲突时哪个规则优先。Amazon S3 将尝试根据所有复制规则复制对象。但是，如果同一目标存储桶有两个或更多规则，则将根据具有最高优先级的规则复制对象。数字越大，优先级越高。
- DeleteMarkerReplication 通过使用值 Enabled 或 Disabled 指示是否复制删除标记。

在目标配置中，您必须提供希望 Amazon S3 将对象复制到其中的存储桶的名称。

以下示例显示针对 V2 规则的最低要求。为了向后兼容，Amazon S3 仍继续支持 XML V1 格式。有关更多信息，请参阅 向后兼容性。

```
...
    <Rule>
        <ID>Rule-1</ID>
        <Status>Enabled-or-Disabled</Status>
        <Filter>
            <Prefix></Prefix>
        </Filter>
        <Priority>integer</Priority>
        <DeleteMarkerReplication>
           <Status>Enabled-or-Disabled</Status>
```

```
        </DeleteMarkerReplication>
        <Destination>
           <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
        </Destination>
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
...
```

您也可以指定其他配置选项。例如，您可能选择了将一个存储类用于对象副本，将另一个存储类用于源对象。

可选：指定筛选条件

要选择规则应用于的对象子集，请添加一个可选筛选条件。可按对象键前缀、对象标签或二者的组合进行筛选。如果您同时按键前缀和对象标签进行筛选，Amazon S3 会使用逻辑 AND 运算符组合这些筛选条件。换句话中，规则会应用于具有特定键前缀和特定标签的对象子集。

根据对象键前缀进行筛选

要指定具有基于对象键前缀的筛选条件的规则，请使用以下代码。只能指定一个前缀。

```
<Rule>
    ...
    <Filter>
        <Prefix>key-prefix</Prefix>
    </Filter>
    ...
</Rule>
...
```

根据对象标签进行筛选

要指定具有基于对象标签的筛选条件的规则，请使用以下代码。可以指定一个或多个对象标签。

```
<Rule>
    ...
    <Filter>
        <And>
            <Tag>
                <Key>key1</Key>
```

```
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
        </And>
    </Filter>
    ...
</Rule>
...
```

使用键前缀和对象标签进行筛选

要指定具有基于键前缀和对象标签组合的规则筛选条件，请使用以下代码。您需要将这些筛选条件包含在 <And> 父元素中。Amazon S3 会执行逻辑 AND 运算以组合这些筛选条件。换句话说，规则会应用于同时具有特定键前缀和特定标签的对象子集。

```
<Rule>
    ...
    <Filter>
        <And>
            <Prefix>key-prefix</Prefix>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
    </Filter>
    ...
</Rule>
...
```

Note

- 如果您指定了一条带有空 <Filter> 元素的规则，则该规则将应用于您存储桶中的所有对象。

- 当您在实时复制中使用基于标签的复制规则时，必须在 PutObject 操作中使用匹配的复制规则标签来标记新对象。否则，将不会复制对象。如果在 PutObject 操作后标记了对象，也不会复制这些对象。

  要复制在 PutObject 操作之后标记的对象，您必须使用 S3 批量复制。有关分批复制的更多信息，请参阅 复制现有对象。

其他目标配置

在目标配置中，指定您希望 Amazon S3 将对象复制到其中的存储桶。您可以设置配置以将对象从一个源存储桶复制到一个或更多个目标存储桶。

```
...
<Destination>
    <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
</Destination>
...
```

您可在 <Destination> 元素中添加以下选项。

主题

- 指定存储类
- 添加多个目标存储桶
- 为涉及多个目标存储桶的每个复制规则指定不同的参数
- 更改副本所有权
- 启用 S3 Replication Time Control
- 复制使用 AWS KMS 通过服务器端加密所创建的对象

指定存储类

您可以为对象副本指定存储类。默认情况下，Amazon S3 使用源对象的存储类来创建对象副本，如以下示例所示。

```
...
<Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
       <StorageClass>storage-class</StorageClass>
```

```
</Destination>
...
```

添加多个目标存储桶

您可以在单个复制配置中添加多个目标存储桶，如下所示。

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET2</Bucket>
    </Destination>
</Rule>
...
```

为涉及多个目标存储桶的每个复制规则指定不同的参数

在单个复制配置中添加多个目标存储桶时，可以为每个复制规则指定不同的参数，如下所示。

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Disabled</Status>
```

```
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled</Status>
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
    <Destination>
       <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET2</Bucket>
    </Destination>
</Rule>
...
```

更改副本所有权

当源存储桶和目标存储桶由不同账户拥有时，您可以将副本的所有权更改为拥有目标存储桶的 AWS 账户。为此，请添加 `AccessControlTranslation` 元素。这个元素将获取值 `Destination`。

```
...
<Destination>
```

```
   <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
   <Account>destination-bucket-owner-account-id</Account>
   <AccessControlTranslation>
       <Owner>Destination</Owner>
   </AccessControlTranslation>
</Destination>
...
```

如果您没有将 AccessControlTranslation 元素添加到复制配置中，则副本由拥有源对象的同一 AWS 账户 拥有。有关更多信息，请参阅 更改副本拥有者。

启用 S3 Replication Time Control

您可以在复制配置中启用 S3 Replication Time Control（S3 RTC）。S3 RTC 在几秒钟内复制大多数对象，并在 15 分钟内复制 99.99% 的对象（由服务等级协议提供支持）。

> **Note**
>
> EventThreshold 和 Time 只接受值 <Minutes>15</Minutes>。

```
...
<Destination>
  <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
  <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
</Destination>
...
```

有关更多信息，请参阅 使用 S3 Replication Time Control（S3 RTC）满足合规性要求。有关 API 示例，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketReplication。

复制使用 AWS KMS 通过服务器端加密所创建的对象

您的源存储桶可能包含一些对象，它们是使用 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）创建的。默认情况下，Amazon S3 不复制这些对象。您可以有选择地指示 Amazon S3 复制这些对象。要做到这一点，首先通过添加 SourceSelectionCriteria 元素显式选择此功能。然后提供 AWS KMS key（对于目标存储桶的 AWS 区域），用于加密对象副本。以下示例显示了如何指定这些元素。

```
...
<SourceSelectionCriteria>
  <SseKmsEncryptedObjects>
    <Status>Enabled</Status>
  </SseKmsEncryptedObjects>
</SourceSelectionCriteria>
<Destination>
  <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
  <EncryptionConfiguration>
    <ReplicaKmsKeyID>AWS KMS key ID to use for encrypting object replicas</
ReplicaKmsKeyID>
  </EncryptionConfiguration>
</Destination>
...
```

有关更多信息，请参阅 复制加密对象（SSE-C、SSE-S3、SSE-KMS、DSSE-KMS）。

复制配置示例

首先，您可以根据需要将以下复制配置示例添加到存储桶。

> **Important**
>
> 要向存储桶添加复制配置，您必须具有 iam:PassRole 权限。此权限允许您传递向 Amazon S3 授予复制权限的 IAM 角色。通过提供复制配置 XML 中 Role 元素中使用的 Amazon 资源名称（ARN），可指定 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的向用户授予将角色传递给 AWS 服务的权限。

Example 1：包含一个规则的复制配置

以下基本复制配置指定一个规则。该规则指定 Amazon S3 可担任的一个 IAM 角色以及用于对象副本的目标存储桶。Enabled 的 Status 值指示该规则是有效的。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>

    <Destination><Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

要选择要复制的对象子集，可添加一个筛选条件。在以下配置中，该筛选条件指定一个对象键前缀。此规则将应用于其键名中带有前缀 *Tax/* 的对象。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
       <Prefix>Tax/</Prefix>
    </Filter>

    <Destination><Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

如果您指定 Filter 元素，则还必须包含 Priority 和 DeleteMarkerReplication 元素。在此示例中，由于只有一个规则，因此 Priority 无关紧要。

在以下配置中，筛选条件指定一个前缀和两个标签。规则将应用于具有指定键前缀和标签的对象子级。具体来说，规则将应用于键名前缀为 *Tax/* 并具有指定对象标签的对象。由于只有一个规则，Priority 不适用。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
        <And>
          <Prefix>Tax/</Prefix>
          <Tag>
             <Tag>
                <Key>tagA</Key>
                <Value>valueA</Value>
             </Tag>
          </Tag>
          <Tag>
             <Tag>
                <Key>tagB</Key>
                <Value>valueB</Value>
             </Tag>
          </Tag>
       </And>

    </Filter>

    <Destination><Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

您可以为对象副本指定存储类，如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket>
```

```
        <StorageClass>storage-class</StorageClass>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

您可以指定 Amazon S3 支持的任何存储类。

Example 2：包含两个规则的复制配置

Example

在以下复制配置中：

- 每个规则按不同的键前缀进行筛选，以便每个规则应用于不同的对象子集。在此示例中，Amazon S3 复制键名为 *Tax/doc1.pdf* 和 *Project/project1.txt* 的对象，但是它不复制键名为 *PersonalDoc/documentA* 的对象。
- 由于规则应用于两个不同的对象子级，因此规则优先级无关紧要。下一示例将说明应用规则优先级时会发生的情况。
- 第二个规则为对象副本指定 S3 Standard-IA 存储类。Amazon S3 对这些对象副本使用指定的存储类。

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Tax</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
     ...
  </Rule>
```

```
 <Rule>
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Project</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
     <StorageClass>STANDARD_IA</StorageClass>
    </Destination>
     ...
  </Rule>


</ReplicationConfiguration>
```

Example 3：包含两个具有重叠前缀的规则的复制配置

在此配置中，两个规则指定具有重叠键前缀 *star/* 和 *starship/* 的筛选条件。两个规则应用于键名称为 *starship-x* 的对象。在此示例中，Amazon S3 使用规则优先级确定要应用哪个规则。数字越大，优先级越高。

```
<ReplicationConfiguration>

  <Role>arn:aws:iam::account-id:role/role-name</Role>

  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>star</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
  </Rule>
```

```
  <Rule>
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>starship</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::DOC-EXAMPLE-BUCKET1</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

Example 4：示例演练

有关演练示例，请参阅 配置实时复制的示例。

有关复制配置的 XML 结构的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketReplication。

向后兼容性

复制配置 XML 的最新版本为 V2。XML V2 复制配置是那些包含规则的 `Filter` 元素和指定 S3 Replication Time Control（S3 RTC）的规则的配置。

要查看复制配置版本，您可以使用 `GetBucketReplication` API 操作。有关更多信息，请参阅 Amazon Simple Storage Service API 参考中的 GetBucketReplication。

为了向后兼容，Amazon S3 仍继续支持 XML V1 复制配置。如果您已使用 XML V1 复制配置，请考虑影响向后兼容性的以下问题：

- 复制配置 XML V2 包含规则的 `Filter` 元素。通过 `Filter` 元素，您可以指定基于对象键前缀和/或标签的对象筛选条件，以确定规则要应用于的范围。复制配置 XML V1 仅支持基于键前缀的筛选。在这种情况下，您直接添加 `Prefix` 作为 `Rule` 元素的子元素，如以下示例所示。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Prefix>key-prefix</Prefix>
```

```
    <Destination><Bucket>arn:aws:s3:::amzn-s3-demo-bucket</Bucket></Destination>

  </Rule>
</ReplicationConfiguration>
```

为了向后兼容，Amazon S3 继续支持 V1 配置。

- 当您从源存储桶中删除对象而不指定对象版本 ID 时，Amazon S3 会添加一个删除标记。如果使用复制配置 XML V1，Amazon S3 会复制因用户操作产生的删除标记。换句话说，Amazon S3 仅在用户删除对象时才复制删除标记。如果 Amazon S3 删除了过期的对象（作为生命周期操作的一部分），Amazon S3 不复制删除标记。

  在 V2 复制配置中，您可以为非基于标签的规则启用删除标记复制。有关更多信息，请参阅 在存储桶之间复制删除标记。

为实时复制设置权限

设置实时复制时，您必须获取必要的权限，如下所示：

- Amazon S3 需要代表您复制对象的权限。您通过创建一个 IAM 角色，然后在复制配置中指定该角色，授予这些权限。
- 当源存储桶与目标存储桶不是由相同亚马逊云科技账户拥有时，目标存储桶的拥有者必须向源存储桶拥有者授予存储副本的权限。

主题

- 创建 IAM 角色
- 在源存储桶和目标存储桶由不同的 AWS 账户 拥有时授予权限
- 授予 S3 批量操作的权限
- 更改副本所有权
- 启用从源存储桶接收复制的对象

创建 IAM 角色

默认情况下，所有 Amazon S3 资源（存储桶、对象和相关子资源）都是私有的，只有资源拥有者可以访问相应的资源。Amazon S3 需要权限才能从源存储桶读取和复制对象。您通过创建一个 IAM 角色并在复制配置中指定该角色，授予这些权限。

此部分介绍信任策略及所需的最低权限策略。示例演练提供创建 IAM 角色的分步说明。有关更多信息，请参阅 配置实时复制的示例。

- 以下示例显示了一个信任策略，您在其中将 Amazon S3 标识为可担任角色的服务主体。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

- 以下示例显示了一个信任策略，您在其中将 Amazon S3 和 S3 批量操作标识为服务主体。如果您正在创建分批复制任务，这很有用。有关更多信息，请参阅 为第一个复制规则或新目标创建分批复制任务。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service": [
              "s3.amazonaws.com",
              "batchoperations.s3.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

有关 IAM 角色的更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

- 以下示例显示了一个访问策略，您在其中向角色授予代表您执行复制任务的权限。当 Amazon S3 担任角色时，它将具有您在此策略中指定的权限。在此策略中，*amzn-s3-demo-bucket1* 是源存储桶，*amzn-s3-demo-bucket2* 是目标存储桶。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-bucket1"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-bucket1/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket2/*"
      }
   ]
}
```

访问策略授予以下操作的权限：

- `s3:GetReplicationConfiguration` 和 `s3:ListBucket` – 针对 *amzn-s3-demo-bucket1* 存储存储桶上这些操作的权限允许 Amazon S3 检索复制配置和列出存储桶内容。（当前权限模型需要 `s3:ListBucket` 权限来访问删除标记）。
- `s3:GetObjectVersionForReplication` 和 `s3:GetObjectVersionAcl` – 针对所有对象授予的这些操作的权限允许 Amazon S3 获取特定对象版本和对象关联的访问控制列表 (ACL)。
- `s3:ReplicateObject` 和 `s3:ReplicateDelete` – 针对 *amzn-s3-demo-bucket2* 存储桶（目标存储桶）中所有对象的这些操作的权限允许 Amazon S3 将对象或删除标记复制到目标存储桶。有关删除标记的信息，请参阅 删除操作对复制操作有何影响。

  > **Note**
  >
  > 对 *amzn-s3-demo-bucket2* 存储桶（目标存储桶）执行 `s3:ReplicateObject` 操作的权限还允许复制元数据，例如对象标签和 ACL。因此，您无需显式授予执行 `s3:ReplicateTags` 操作的权限。

- `s3:GetObjectVersionTagging` – 针对 *amzn-s3-demo-bucket1* 存储桶（源存储桶）中对象的此操作的权限允许 Amazon S3 读取复制的对象标签。有关更多信息，请参阅 使用标签对存储进行分类。如果 Amazon S3 没有这些权限，它将复制对象而不是对象标签。

有关 Amazon S3 操作的列表，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

> **Important**
>
> 拥有 IAM 角色的 AWS 账户 必须拥有其向 IAM 角色授予的操作的权限。
> 例如，假定源存储桶包含由另一个 AWS 账户 拥有的对象。对象的拥有者必须通过对象 ACL 向拥有 IAM 角色的 AWS 账户 显式授予必要的权限。否则，Amazon S3 无法访问这些对象，因而这些对象的复制将失败。有关 ACL 权限的信息，请参阅 访问控制列表 (ACL) 概述。
> 此处介绍的权限与最低复制配置相关。如果您选择添加可选复制配置，则必须向 Amazon S3 授予其他权限。

在源存储桶和目标存储桶由不同的 AWS 账户 拥有时授予权限

当源存储桶和目标存储桶由不同的账户拥有时，目标存储桶的拥有者还必须添加一个存储桶策略，以向源存储桶的拥有者授予执行复制操作的权限，如下所示。在此策略中，*amzn-s3-demo-bucket2* 是目标存储桶。

> **Note**
>
> 角色的 ARN 格式可能看起来不同。如果使用控制台创建角色，则 ARN 格式为 arn:aws:iam::*account-ID*:role/service-role/*role-name*。如果使用 AWS CLI 创建角色，则 ARN 格式为 arn:aws:iam::*account-ID*:role/*role-name*。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

```
{
   "Version":"2012-10-17",
   "Id":"PolicyForDestinationBucket",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3:ReplicateDelete",
            "s3:ReplicateObject"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket2/*"
      },
      {
         "Sid":"Permissions on bucket",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action": [
            "s3:List*",
            "s3:GetBucketVersioning",
            "s3:PutBucketVersioning"
```

```
         ],
         "Resource":"arn:aws:s3::amzn-s3-demo-bucket2"
      }
   ]
}
```

有关示例，请参阅当源存储桶和目标存储桶由不同账户拥有时配置复制。

如果向源存储桶中的对象添加标签，请注意以下事项：

- 如果源存储桶拥有者向 Amazon S3 授予 s3:GetObjectVersionTagging 和 s3:ReplicateTags 操作的权限来复制对象标签（通过 IAM 角色），则 Amazon S3 将复制标签以及对象。有关 IAM 角色的信息，请参阅 创建 IAM 角色。
- 如果目标存储桶的所有者不希望复制标签，则他们可以向目标存储桶策略添加以下语句来显式拒绝 s3:ReplicateTags 操作的权限。在此策略中，*amzn-s3-demo-bucket2* 是目标存储桶。

```
...
   "Statement":[
      {
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-id:role/service-role/source-
account-IAM-role"
         },
         "Action":"s3:ReplicateTags",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket2/*"
      }
   ]
...
```

授予 S3 批量操作的权限

S3 批处理复制为您提供了一种方法，以复制在复制配置实施之前就已存在的对象、之前已复制的对象和复制失败的对象。在新复制配置中创建第一个规则或通过 AWS Management Console 将新目标添加到现有配置时，您可以创建一次性批量复制任务。您也可以通过创建批量操作任务来启动现有复制配置的批量复制。

有关批量复制 IAM 角色和策略示例，请参阅配置 IAM 策略以进行分批复制。

更改副本所有权

当源存储桶和目标存储桶由不同的 AWS 账户 拥有时，您可以指示 Amazon S3 将副本的所有权更改为拥有目标存储桶的 AWS 账户。有关拥有者覆盖的更多信息，请参阅更改副本拥有者。

启用从源存储桶接收复制的对象

您可以快速生成所需的策略，以便通过 AWS Management Console从源存储桶接收复制的对象。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，选择要用作目标存储桶的存储桶。
4. 选择 Management（管理）选项卡，然后向下滚动到 Replication rules（复制规则）。
5. 对于 Actions（操作），选择 Receive replicated objects（接收复制的对象）。

   按照提示进行操作并输入源存储桶账户的 AWS 账户 ID，然后选择 Generate policies（生成策略）。这将生成 Amazon S3 存储桶策略和 KMS 密钥策略。
6. 要将此策略添加到现有存储桶策略中，请选择 Apply settings（应用设置），或者选择 Copy（复制）以手动复制更改。
7. （可选）在 AWS Key Management Service 控制台上将 AWS KMS 策略复制到所需的 KMS 密钥策略。

配置实时复制的示例

以下各示例介绍如何为常见使用案例配置实时复制。

> **Note**
>
> 实时复制指同区域复制 (SRR) 和跨区域复制 (CRR)。实时复制不会复制在设置复制之前就存在于存储桶中的任何对象。想要复制在设置复制之前就存在的对象，请使用按需复制。要同步存储桶和按需复制现有对象，请参阅复制现有对象。

这些示例演示了如何使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK（展示了AWS SDK for Java 和AWS SDK for .NET 示例）创建复制配置。

有关安装和配置 AWS CLI 的信息，请参阅 AWS Command Line Interface 用户指南中的以下主题。

- 安装 AWS Command Line Interface
- 配置 AWS CLI – 您必须设置至少一个配置文件。如果要探讨跨账户方案，请设置两个配置文件。

有关 AWS SDK 的信息，请参阅适用于 Java 的 AWS 开发工具包和适用于 .NET 的 AWS 开发工具包。

> **Tip**
>
> 有关演示如何使用实时复制来复制数据的分步教程，请参阅教程：使用 S3 复制在 AWS 区域内和区域之间复制数据。

主题

- 为同一账户拥有的源存储桶和目标存储桶配置复制
- 当源存储桶和目标存储桶由不同账户拥有时配置复制
- 使用 S3 Replication Time Control（S3 RTC）满足合规性要求
- 复制加密对象（SSE-C、SSE-S3、SSE-KMS、DSSE-KMS）
- 使用 Amazon S3 副本修改同步复制元数据更改
- 在存储桶之间复制删除标记

为同一账户拥有的源存储桶和目标存储桶配置复制

复制是在相同或跨不同 AWS 区域 的存储桶自动、异步地复制对象。复制操作会将源存储桶中新创建的对象和对象更新复制到目标存储桶。有关更多信息，请参阅 复制对象概述。

配置复制时，需要向源存储桶添加复制规则。复制规则定义要复制的源存储桶对象和存储已复制对象的目标存储桶。您可以创建一条规则，以复制存储桶中的所有对象或具有特定键名前缀和/或一个或多个对象标签的对象子集。目标存储桶与源存储桶可以位于同一 AWS 账户 中，也可以位于不同的账户中。

如果您指定要删除的对象版本 ID，Amazon S3 会在源存储桶中删除该对象版本。但不会将删除操作复制到目标存储桶中。换句话说，它不会从目标存储桶中删除同一对象版本。这会防止恶意删除数据。

当您将复制规则添加到存储桶后，默认情况下将启用复制规则，使该规则在您保存它后立即启动。

在此示例中，您为由同一 AWS 账户 拥有的源存储桶和目标存储桶设置复制。提供了使用 Amazon S3 控制台、AWS Command Line Interface (AWS CLI) 以及 AWS SDK for Java 和 AWS SDK for .NET 的示例。

使用 S3 控制台

要在目标存储桶与源存储桶位于同一 AWS 账户中时配置复制规则，请按照以下步骤操作。

如果目标存储桶与源存储桶位于不同的账户中，您必须向目标存储桶添加存储桶策略以便为源存储桶账户的所有者授予复制目标存储桶中的对象的权利。有关更多信息，请参阅 在源存储桶和目标存储桶由不同的 AWS 账户 拥有时授予权限。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，请选择您想要的存储桶的名称。
4. 选择管理选项卡，向下滚动到复制规则，然后选择创建复制规则。
5. 在复制规则配置部分的复制规则名称下，输入规则的名称，以帮助以后识别该规则。该名称是必填项，并且它在存储桶内必须是唯一的。
6. 在状态下，已启用默认情况下处于选中状态。已启用的规则将在您保存它后立即开始工作。如果您想以后再启用该规则，请选择已禁用。
7. 如果存储桶具有现有的复制规则，系统会指示您为规则设置优先级。必须为规则设置优先级，以避免因在多个规则的范围内包含对象而引起冲突。如果规则重叠，Amazon S3 会使用规则优先级确定要应用哪个规则。数字越大，优先级越高。有关规则优先级的更多信息，请参阅 复制配置。
8. 在源存储桶下，您可以通过以下选项设置复制源：
   - 要复制整个存储桶，请选择应用到存储桶中的所有对象。
   - 要复制具有相同前缀的所有对象，请选择 Limit the scope of this rule using one or more filters (使用一个或多个筛选条件限制此规则的范围)。这会将复制限制为名称以您指定的前缀（例如 `pictures`）开头的所有对象。在前缀框中输入前缀。

     > **Note**
     >
     > 如果您输入的前缀是文件夹名称，则必须使用 /（正斜杠）作为最后一个字符（例如，`pictures/`）。

- 要复制具有一个或多个对象标签的所有对象，请选择添加标签，然后在框中输入键值对。重复上述过程以添加其他标签。您可以组合前缀和标签。有关对象标签的更多信息，请参阅 使用标签对存储进行分类。

新的复制配置 XML 模式支持前缀和标签筛选以及规则优先级划分。有关新架构的更多信息，请参阅 向后兼容性。如需详细了解在用户界面后台工作的 Amazon S3 API 所使用的 XML，请参阅 复制配置。新架构被描述为复制配置 XML V2。

9. 在目标下，选择您希望 Amazon S3 将对象复制到的存储桶。

> **Note**
>
> 目标存储桶的数量仅限于给定分区中的 AWS 区域 的数量。分区是一组区域。AWS 目前有三个分区：aws（标准区域）、aws-cn（中国区域）和 aws-us-gov（AWS GovCloud (US) 区域）。要请求提高目标存储桶限额，您可以使用服务限额。

- 要复制到您的账户中的一个或多个存储桶，请选取选择此账户中的存储桶，然后输入或浏览目标存储桶。
- 要复制到其他 AWS 账户中的一个或多个存储存储桶，请选择指定另一个账户中的存储存储桶，然后输入目标存储存储桶账户 ID 和存储桶名称。

  如果目标存储桶与源存储桶位于不同的账户中，您必须向目标存储桶添加存储桶策略以便为源存储桶账户的所有者授予复制对象的权限。有关更多信息，请参阅 在源存储桶和目标存储桶由不同的 AWS 账户 拥有时授予权限。

  （可选）如果要帮助标准化目标存储桶中的新对象的所有权，请选择将对象所有权更改为目标存储桶拥有者。有关此选项的更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

> **Note**
>
> 如果未对目标存储桶启用版本控制，您将收到包含 Enable versioning（启用版本控制）按钮的警告。请选择此按钮可对存储桶启用版本控制。

10. 设置 AWS Identity and Access Management (IAM) 角色，Amazon S3 可以代入该角色以代表您复制对象。

    要设置 IAM 角色，请在 IAM 角色部分中，从 IAM 角色下拉列表中选择以下任一选项：

- 我们强烈建议您选择 Create new role (创建新角色)，让 Amazon S3 为您创建一个新的 IAM 角色。当您保存该规则后，将为 IAM 角色生成一个与您选择的源和目标存储桶匹配的新策略。
- 您可以选择使用现有的 IAM 角色。在这种情况下，您必须选择一个角色，该角色会授予 Amazon S3 必要的权限以进行复制。如果该角色未按照您的复制规则授予 Amazon S3 足够的权限，复制会失败。

> **Important**
>
> 将复制规则添加到存储桶时，您必须具有 `iam:PassRole` 权限才能传递授予 Amazon S3 复制权限的 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的向用户授予将角色传递给 AWS 服务的权限。

11. 要复制源存储桶中使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）进行加密的对象，请在加密下选择复制已使用 AWS KMS 加密的对象。在用于加密目标对象的 AWS KMS 密钥下，是您允许复制使用的源密钥。预设情况下，所有源 KMS 密钥都包含在内。要缩小 KMS 密钥选择范围，可以选择别名或密钥 ID。

    使用您未选择的 AWS KMS keys 加密的对象不会进行复制。系统为您选择一个 KMS 密钥或一组 KMS 密钥，您可以自行选择 KMS 密钥。有关将 AWS KMS 用于复制的信息，请参阅 复制加密对象（SSE-C、SSE-S3、SSE-KMS、DSSE-KMS）。

    > **Important**
    >
    > 在复制使用 AWS KMS 加密的对象时，AWS KMS 请求速率会在源区域中加倍并在目标区域中增加相同的量。之所以对 AWS KMS 的调用率会增加，是因为数据是使用您为复制目标区域定义的 KMS 密钥进行重新加密的。对于每个区域的每个调用账户，AWS KMS 设定了一个请求速率限额。有关限额默认值的信息，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS 限额 – 每秒请求数：因情况而异。
    > 如果您在复制期间的当前 Amazon S3 PUT 对象请求速率超过您账户的默认 AWS KMS 速率限制的一半，则建议您请求提高您的 AWS KMS 请求速率限额。要请求提高，请在联系我们处的 AWS Support 中心内创建一个案例。例如，假设您当前的 PUT 对象请求速率为每秒 1000 个请求并且您使用 AWS KMS 加密对象。在此情况下，建议您让 AWS Support 将您在源区域和目标区域中的 AWS KMS 速率限制提高到每秒 2500 个请求（如果不同），以确保不受 AWS KMS 的限制。

要查看源存储桶中的 PUT 对象请求速率，请查看 Amazon S3 的 Amazon CloudWatch 请求指标中的 `PutRequests`。有关查看 CloudWatch 指标的信息，请参阅使用 S3 控制台

如果选择复制使用 AWS KMS 加密的对象，请执行以下操作：

- 在用于加密目标对象的 AWS KMS key 下，通过以下方式之一指定您的 KMS 密钥：
  - 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 密钥中进行选择，并从可用密钥的列表中选择您的 KMS 密钥。

    AWS 托管式密钥 (`aws/s3`) 和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。
  - 要输入 KMS 密钥 Amazon 资源名称（ARN），请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。这会加密目标存储桶中的副本。您可以在 IAM 控制台中的加密密钥下方找到您的 KMS 密钥的 ARN。
  - 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

    有关创建 AWS KMS key 的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的创建密钥。

> **Important**
>
> 您只能使用与存储桶所在相同 AWS 区域 中启用的 KMS 密钥。选择从您的 KMS 密钥中选择时，S3 控制台对于每个区域仅列出 100 个 KMS 密钥。如果您在同一区域中有超过 100 个 KMS 密钥，您只会在 S3 控制台中看到前 100 个 KMS 密钥。要使用控制台中未列出的 KMS 密钥，请选择输入 AWS KMS key ARN，然后输入您的 KMS 密钥 ARN。
>
> 在 Amazon S3 中使用 AWS KMS key 进行服务器端加密时，您必须选择对称加密 KMS 密钥。Amazon S3 仅支持对称加密 KMS 密钥，而不支持非对称 KMS 密钥。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的确定对称和非对称 KMS 密钥。

有关创建 AWS KMS key 的更多信息，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。有关将 AWS KMS 与 Amazon S3 结合使用的更多信息，请参阅使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。

12. 在目标存储类下，如果要将数据复制到目标中的特定存储类，请选择更改复制对象的存储类。然后选择要用于目标中的已复制对象的存储类。如果您不选择此选项，已复制对象的存储类将与原始对象的类相同。
13. 设置其他复制选项时，您具有以下附加选项：

    - 如果要在复制配置中启用 S3 Replication Time Control（S3 RTC），请选择复制时间控制（RTC）。有关此选项的更多信息，请参阅 使用 S3 Replication Time Control（S3 RTC）满足合规性要求。
    - 如果要在复制配置中启用 S3 复制指标，请选择 Replication metrics and events（复制指标和事件）。有关更多信息，请参阅使用复制指标和 S3 事件通知监控进度。
    - 如果要在复制配置中启用删除标记复制，请选择 Delete marker replication (删除标记复制)。有关更多信息，请参阅在存储桶之间复制删除标记。
    - 如果要在复制配置中启用 Amazon S3 副本修改同步，请选择 Replica modification sync (副本修改同步)。有关更多信息，请参阅使用 Amazon S3 副本修改同步复制元数据更改。

    > **Note**
    >
    > 使用 S3 RTC 或 S3 复制指标时，需要支付额外费用。

14. 要完成，请选择 Save (保存)。
15. 在您保存规则之后，可以通过选择您的规则并选择 Edit rule (编辑规则) 来编辑、启用、禁用或删除您的规则。

使用 AWS CLI

要使用 AWS CLI 在源存储桶和目标存储桶由同一 AWS 账户 拥有时设置复制，请执行以下操作：

- 创建源存储桶和目标存储桶
- 对存储桶启用版本控制功能
- 创建为 Amazon S3 授予复制对象权限的 IAM 角色
- 将复制配置添加到源存储桶中

要验证您的设置，请对其进行测试。

当源存储桶和目标存储桶由同一 AWS 账户 拥有时设置复制

1. 为 AWS CLI 设置凭证配置文件。在此示例中，我们使用配置文件名称 `acctA`。有关设置凭证配置文件的更多信息，请参阅《AWS Command Line Interface 用户指南》中的命名配置文件。

   > **Important**
   >
   > 用于本练习的配置文件必须具有必要的权限。例如，在复制配置中，指定 Amazon S3 可担任的 IAM 角色。仅当您所使用的配置文件具有 `iam:PassRole` 权限时，您才能执行此操作。有关更多信息，请参阅 IAM 用户指南中的向用户授予将角色传递给 AWS 服务的权限。如果您使用管理员凭证创建命名配置文件，则可执行所有任务。

2. 创建一个 *`source`* 存储桶，并对其启用版本控制。以下代码在美国东部（弗吉尼亚州北部）（us-east-1）区域中创建一个 *`source`* 存储桶。

```
aws s3api create-bucket \
--bucket source \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket source \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. 创建一个 *`destination`* 存储桶，并对其启用版本控制。以下代码在美国西部（俄勒冈州）（us-west-2）区域中创建一个 *`destination`* 存储桶。

   > **Note**
   >
   > 要在源存储桶和目标存储桶位于同一 AWS 账户 时设置复制配置，请使用同一配置文件。此示例使用 `acctA`。要在两个存储桶由不同 AWS 账户 拥有时对复制配置进行测试，您需要为每个存储桶指定不同的配置文件。在此示例中，我们对目标存储桶使用 `acctB` 配置文件。

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. 创建一个 IAM 角色。您将在稍后添加到#存储桶的复制配置中指定此角色。Amazon S3 担任此角色以代表您复制对象。分两个步骤创建 IAM 角色：

   - 创建角色。
   - 将权限策略附加到角色。

   a. 创建 IAM 角色。

      i. 复制以下信任策略，并将其保存到本地计算机上当前目录中一个名为 s3-role-trust-policy.json 的文件。此策略向 Amazon S3 服务主体授予担任该角色的权限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. 运行以下命令以创建角色。

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
```

```
--profile acctA
```

b. 将权限策略附加到角色。

i. 复制以下权限策略，并将其保存到本地计算机上当前目录中一个名为 `s3-role-permissions-policy.json` 的文件。此策略授予对各种 Amazon S3 存储桶和对象操作的权限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Resource":"arn:aws:s3:::destination-bucket/*"
      }
   ]
}
```

ii. 运行以下命令以创建策略并将其附加到角色。

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-permissions-policy.json \
--policy-name replicationRolePolicy \
--profile acctA
```

5. 将复制配置添加到 *source* 存储桶中。

a. 尽管 Amazon S3 API 要求复制配置采用 XML 形式，但 AWS CLI 要求您以 JSON 形式指定复制配置。将以下 JSON 保存到计算机上本地目录中一个名为 `replication.json` 文件。

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket": "arn:aws:s3:::destination-bucket"
      }
    }
  ]
}
```

b. 通过提供 *destination-bucket* 和 *IAM-role-ARN* 的值来更新 JSON。保存更改。

c. 运行以下命令以将复制配置添加到源存储桶中。请务必提供 *source* 存储桶名称。

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
--profile acctA
```

要检索复制配置，请使用 `get-bucket-replication` 命令。

```
$ aws s3api get-bucket-replication \
--bucket source \
```

```
--profile acctA
```

6. 在 Amazon S3 控制台中测试该设置：

   a. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

   b. 在 *source* 存储桶中，创建一个名为 Tax 的文件夹。

   c. 将示例对象添加到 *source* 存储桶中的 Tax 文件夹。

> **Note**
>
> Amazon S3 复制对象所需的时间量取决于对象大小。有关如何查看复制状态的信息，请参阅获取复制状态信息。

在 *destination* 存储桶中，确认以下几点：

- 该 Amazon S3 已复制对象。
- 在对象 properties (属性) 中，Replication Status (复制状态) 已设置为 Replica（将其标识为副本对象）。
- 在对象 properties (属性) 中，权限部分未显示权限。这意味着副本仍由 *source* 存储桶拥有者拥有，而 *destination* 存储桶拥有者不具有对象副本的权限。您可以添加可选配置以指示 Amazon S3 更改副本所有权。有关示例，请参阅 如何更改副本拥有者。

使用 AWS SDK

以下代码示例分别使用 AWS SDK for Java 和 AWS SDK for .NET 向存储桶添加复制配置。

Java

以下示例向存储桶添加复制配置，然后检索并验证该配置。有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.identitymanagement.AmazonIdentityManagement;
import
 com.amazonaws.services.identitymanagement.AmazonIdentityManagementClientBuilder;
import com.amazonaws.services.identitymanagement.model.CreateRoleRequest;
import com.amazonaws.services.identitymanagement.model.PutRolePolicyRequest;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.BucketReplicationConfiguration;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.CreateBucketRequest;
import com.amazonaws.services.s3.model.DeleteMarkerReplication;
import com.amazonaws.services.s3.model.DeleteMarkerReplicationStatus;
import com.amazonaws.services.s3.model.ReplicationDestinationConfig;
import com.amazonaws.services.s3.model.ReplicationRule;
import com.amazonaws.services.s3.model.ReplicationRuleStatus;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;
import com.amazonaws.services.s3.model.StorageClass;
import com.amazonaws.services.s3.model.replication.ReplicationFilter;
import com.amazonaws.services.s3.model.replication.ReplicationFilterPredicate;
import com.amazonaws.services.s3.model.replication.ReplicationPrefixPredicate;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CrossRegionReplication {

        public static void main(String[] args) throws IOException {
```

```
                Regions clientRegion = Regions.DEFAULT_REGION;
                String accountId = "*** Account ID ***";
                String roleName = "*** Role name ***";
                String sourceBucketName = "*** Source bucket name ***";
                String destBucketName = "*** Destination bucket name ***";
                String prefix = "Tax/";

                String roleARN = String.format("arn:aws:iam::%s:%s", accountId,
roleName);
                String destinationBucketARN = "arn:aws:s3:::" + destBucketName;

                AmazonS3 s3Client = AmazonS3Client.builder()
                                .withCredentials(new ProfileCredentialsProvider())
                                .withRegion(clientRegion)
                                .build();

                createBucket(s3Client, clientRegion, sourceBucketName);
                createBucket(s3Client, clientRegion, destBucketName);
                assignRole(roleName, clientRegion, sourceBucketName,
destBucketName);

                try {

                        // Create the replication rule.
                        List<ReplicationFilterPredicate> andOperands = new
ArrayList<ReplicationFilterPredicate>();
                        andOperands.add(new ReplicationPrefixPredicate(prefix));

                        Map<String, ReplicationRule> replicationRules = new
HashMap<String, ReplicationRule>();
                        replicationRules.put("ReplicationRule1",
                                        new ReplicationRule()
                                                        .withPriority(0)

.withStatus(ReplicationRuleStatus.Enabled)

.withDeleteMarkerReplication(
                                                                        new
DeleteMarkerReplication().withStatus(

   DeleteMarkerReplicationStatus.DISABLED))
                                                        .withFilter(new
ReplicationFilter().withPredicate(
```

```
                                                                    new
ReplicationPrefixPredicate(prefix)))
                                                        .withDestinationConfig(new
ReplicationDestinationConfig()

.withBucketARN(destinationBucketARN)

.withStorageClass(StorageClass.Standard)));

                        // Save the replication rule to the source bucket.
                        s3Client.setBucketReplicationConfiguration(sourceBucketName,
                                        new BucketReplicationConfiguration()
                                                        .withRoleARN(roleARN)

.withRules(replicationRules));

                        // Retrieve the replication configuration and verify that
the configuration
                        // matches the rule we just set.
                        BucketReplicationConfiguration replicationConfig = s3Client

.getBucketReplicationConfiguration(sourceBucketName);
                        ReplicationRule rule =
replicationConfig.getRule("ReplicationRule1");
                        System.out.println("Retrieved destination bucket ARN: "
                                        +
rule.getDestinationConfig().getBucketARN());
                        System.out.println("Retrieved priority: " +
rule.getPriority());
                        System.out.println("Retrieved source-bucket replication rule
status: " + rule.getStatus());
                } catch (AmazonServiceException e) {
                        // The call was transmitted successfully, but Amazon S3
couldn't process
                        // it, so it returned an error response.
                        e.printStackTrace();
                } catch (SdkClientException e) {
                        // Amazon S3 couldn't be contacted for a response, or the
client
                        // couldn't parse the response from Amazon S3.
                        e.printStackTrace();
                }
        }
```

```
        private static void createBucket(AmazonS3 s3Client, Regions region, String
 bucketName) {
                CreateBucketRequest request = new CreateBucketRequest(bucketName,
 region.getName());
                s3Client.createBucket(request);
                BucketVersioningConfiguration configuration = new
 BucketVersioningConfiguration()
                                .withStatus(BucketVersioningConfiguration.ENABLED);

                SetBucketVersioningConfigurationRequest enableVersioningRequest =
 new SetBucketVersioningConfigurationRequest(
                                bucketName, configuration);
                s3Client.setBucketVersioningConfiguration(enableVersioningRequest);

        }

        private static void assignRole(String roleName, Regions region, String
 sourceBucket, String destinationBucket) {
                AmazonIdentityManagement iamClient =
 AmazonIdentityManagementClientBuilder.standard()
                                .withRegion(region)
                                .withCredentials(new ProfileCredentialsProvider())
                                .build();
                StringBuilder trustPolicy = new StringBuilder();
                trustPolicy.append("{\\r\\n   ");
                trustPolicy.append("\\\"Version\\\":\\\"2012-10-17\\\",\\r\\n   ");
                trustPolicy.append("\\\"Statement\\\":[\\r\\n      {\\r\\n
 ");
                trustPolicy.append("\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Principal\\\":{\\r\\n            ");
                trustPolicy.append("\\\"Service\\\":\\\"s3.amazonaws.com\\\"\\r\\n
       },\\r\\n         ");
                trustPolicy.append("\\\"Action\\\":\\\"sts:AssumeRole\\\"\\r\\n
  }\\r\\n   ]\\r\\n}");

                CreateRoleRequest createRoleRequest = new CreateRoleRequest()
                                .withRoleName(roleName)

 .withAssumeRolePolicyDocument(trustPolicy.toString());

                iamClient.createRole(createRoleRequest);

                StringBuilder permissionPolicy = new StringBuilder();
                permissionPolicy.append(
```

```
                                "{\\r\\n   \\\"Version\\\":\\\"2012-10-17\\\",\\r\\n
   \\\"Statement\\\":[\\r\\n      {\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Action\\\":[\\r\\n             ");
                permissionPolicy.append("\\\"s3:GetObjectVersionForReplication\\\",\
\r\\n            ");
                permissionPolicy.append(
                                "\\\"s3:GetObjectVersionAcl\\\"\\r\\n         ],\\r\
\n         \\\"Resource\\\":[\\r\\n            ");
                permissionPolicy.append("\\\"arn:aws:s3:::");
                permissionPolicy.append(sourceBucket);
                permissionPolicy.append("/*\\\"\\r\\n         ]\\r\\n      },\\r\\n
     {\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Effect\\\":\\\"Allow\\\",\\r\\n         \\
\"Action\\\":[\\r\\n            ");
                permissionPolicy.append(
                                "\\\"s3:ListBucket\\\",\\r\\n            \\
\"s3:GetReplicationConfiguration\\\"\\r\\n         ");
                permissionPolicy.append("],\\r\\n         \\\"Resource\\\":[\\r\\n
          \\\"arn:aws:s3:::");
                permissionPolicy.append(sourceBucket);
                permissionPolicy.append("\\r\\n         ");
                permissionPolicy
                                .append("]\\r\\n      },\\r\\n      {\\r\\n
 \\\"Effect\\\":\\\"Allow\\\",\\r\\n         ");
                permissionPolicy.append(
                                "\\\"Action\\\":[\\r\\n            \\
\"s3:ReplicateObject\\\",\\r\\n            ");
                permissionPolicy
                                .append("\\\"s3:ReplicateDelete\\\",\\r\\n
  \\\"s3:ReplicateTags\\\",\\r\\n            ");
                permissionPolicy.append("\\\"s3:GetObjectVersionTagging\\\"\\r\\n\\r
\\n         ],\\r\\n         ");
                permissionPolicy.append("\\\"Resource\\\":\\\"arn:aws:s3:::");
                permissionPolicy.append(destinationBucket);
                permissionPolicy.append("/*\\\"\\r\\n      }\\r\\n   ]\\r\\n}");

                PutRolePolicyRequest putRolePolicyRequest = new
 PutRolePolicyRequest()
                                .withRoleName(roleName)
                                .withPolicyDocument(permissionPolicy.toString())
                                .withPolicyName("crrRolePolicy");
```

```
                iamClient.putRolePolicy(putRolePolicyRequest);

        }
}
```

C#

以下 AWS SDK for .NET 代码示例向存储桶添加复制配置，然后检索该配置。要使用此代码，请为存储桶提供名称，并为 IAM 角色提供 Amazon 资源名称（ARN）。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CrossRegionReplicationTest
    {
        private const string sourceBucket = "*** source bucket ***";
        // Bucket ARN example - arn:aws:s3:::destinationbucket
        private const string destinationBucketArn = "*** destination bucket ARN
 ***";
        private const string roleArn = "*** IAM Role ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint sourceBucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(sourceBucketRegion);
            EnableReplicationAsync().Wait();
        }
        static async Task EnableReplicationAsync()
        {
            try
            {
                ReplicationConfiguration replConfig = new ReplicationConfiguration
                {
```

```
                    Role = roleArn,
                    Rules =
                        {
                            new ReplicationRule
                            {
                                Prefix = "Tax",
                                Status = ReplicationRuleStatus.Enabled,
                                Destination = new ReplicationDestination
                                {
                                    BucketArn = destinationBucketArn
                                }
                            }
                        }
                };

                PutBucketReplicationRequest putRequest = new
PutBucketReplicationRequest
                {
                    BucketName = sourceBucket,
                    Configuration = replConfig
                };

                PutBucketReplicationResponse putResponse = await
s3Client.PutBucketReplicationAsync(putRequest);

                // Verify configuration by retrieving it.
                await RetrieveReplicationConfigurationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }
        private static async Task RetrieveReplicationConfigurationAsync(IAmazonS3
client)
        {
            // Retrieve the configuration.
            GetBucketReplicationRequest getRequest = new GetBucketReplicationRequest
```

```
            {
                BucketName = sourceBucket
            };
            GetBucketReplicationResponse getResponse = await
 client.GetBucketReplicationAsync(getRequest);
            // Print.
            Console.WriteLine("Printing replication configuration information...");
            Console.WriteLine("Role ARN: {0}", getResponse.Configuration.Role);
            foreach (var rule in getResponse.Configuration.Rules)
            {
                Console.WriteLine("ID: {0}", rule.Id);
                Console.WriteLine("Prefix: {0}", rule.Prefix);
                Console.WriteLine("Status: {0}", rule.Status);
            }
        }
    }
}
```

当源存储桶和目标存储桶由不同账户拥有时配置复制

在#存储桶和##存储桶由不同 AWS 账户拥有时设置复制的操作，与这两个存储桶由相同账户拥有时设置复制的操作类似。唯一区别在于，##存储桶拥有者必须通过添加存储桶策略，向#存储桶拥有者授予复制对象的权限。

有关在跨账户场景中使用具有 AWS Key Management Service 的服务器端加密配置复制的更多信息，请参阅为跨账户方案授予其他权限。

当源存储桶和目标存储桶由不同 AWS 账户 拥有时配置复制

1. 在此示例中，您将在两个不同的 AWS 账户 中创建####和##存储桶。您需要为 AWS CLI 设置两个凭证配置文件（在此示例中，我们使用 `acctA` 和 `acctB` 作为配置文件名称）。有关设置凭证配置文件的更多信息，请参阅《AWS Command Line Interface 用户指南》中的命名配置文件。
2. 按照 针对同一账户中的存储桶进行配置 中的分步说明操作，进行以下更改：
   - 对于与#存储桶活动相关的所有 AWS CLI 命令（用于创建#存储桶、启用版本控制和创建 IAM 角色），使用 `acctA` 配置文件。使用 `acctB` 配置文件创建##存储桶。
   - 确保权限策略指定为此示例创建的####和##存储桶。
3. 在控制台中，在##存储存储桶上添加以下存储桶策略，以允许#存储存储桶的拥有者复制对象。请务必编辑该策略，即提供#存储桶拥有者的 AWS 账户 ID 和##存储桶名称。

> **Note**
>
> 要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。将 *DOC-EXAMPLE-BUCKET* 替换为您的存储桶的名称。将 *source-bucket-acct-ID:role/service-role/source-acct-IAM-role* 替换为用于此复制配置的角色。
> 如果您手动创建 IAM 服务角色，请将角色路径设置为 `role/service-role/`，如以下策略示例所示。有关更多信息，请参阅《IAM 用户指南》中的 IAM ARN。

```
{
   "Version":"2012-10-17",
   "Id":"",
   "Statement":[
      {
         "Sid":"Set-permissions-for-objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-acct-ID:role/service-role/source-acct-IAM-role"
         },
         "Action":["s3:ReplicateObject", "s3:ReplicateDelete"],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      },
      {
         "Sid":"Set permissions on bucket",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-acct-ID:role/service-role/source-acct-IAM-role"
         },
         "Action":["s3:GetBucketVersioning", "s3:PutBucketVersioning"],
         "Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET"
      }
   ]
}
```

请选择存储桶并添加存储存储桶策略。有关说明，请参阅 使用 Amazon S3 控制台添加存储桶策略。

在复制中，默认情况下，源对象的拥有者拥有副本。当源存储桶和目标存储桶由不同的 AWS 账户拥有时，您可以添加可选的配置设置以将副本所有权更改为拥有目标存储桶的 AWS 账户。这包括授予 `ObjectOwnerOverrideToBucketOwner` 权限。有关更多信息，请参阅 更改副本拥有者。

更改副本拥有者

在复制中，默认情况下，源对象的拥有者也拥有副本。当源存储桶和目标存储桶由不同的 AWS 账户拥有时，并且您想要将副本拥有权更改为拥有目标存储桶的 AWS 账户，您可以添加可选的配置设置以将副本所有权更改为拥有目标存储桶的 AWS 账户。例如，您可能会执行此操作来限制对对象副本的访问权。这称作复制配置的拥有者覆盖 选项。有关所有者覆盖选项的更多信息，请参阅 向复制配置添加拥有者覆盖选项。有关设置复制配置的信息，请参阅复制对象概述。

要配置拥有者覆盖，您需要执行以下任务：

- 将拥有者覆盖选项添加到复制配置以指示 Amazon S3 更改副本所有权。
- 向 Amazon S3 授予更改副本所有权的权限。
- 在目标存储桶策略中添加允许更改副本所有权的权限。这将允许目标存储桶的拥有者接受对象副本的所有权。

有关更多信息，请参阅 向复制配置添加拥有者覆盖选项。有关带分步说明的有效示例，请参阅如何更改副本拥有者。

对象所有权的强制存储桶拥有者设置

当您使用 Amazon S3 复制，且源和目标存储桶由不同的 AWS 账户 拥有时，目标存储桶的存储桶拥有者可以禁用 ACL（通过对象所有权的强制存储桶拥有者设置）以将副本拥有权更改为拥有目标存储桶的 AWS 账户。此设置模拟现有的所有者覆盖行为，而无需 `s3:ObjectOwnerOverrideToBucketOwner` 权限。这意味着，在强制存储桶拥有者设置的情况下已复制到目标存储桶的所有对象都归目标存储桶拥有者所有。有关对象所有权的更多信息，请参阅为您的存储桶控制对象所有权和禁用 ACL。 。

向复制配置添加拥有者覆盖选项

> **Warning**
>
> 仅在源存储桶和目标存储桶由不同 AWS 账户拥有时添加拥有者覆盖选项。Amazon S3 不会检查存储桶由相同账户还是不同账户拥有。如果您在两个存储桶均由同一 AWS 账户拥有时添加拥有者覆盖，Amazon S3 会应用拥有者覆盖。它向目标存储桶的拥有者授予完全权限，不会将

后续的更新复制到源对象访问控制列表（ACL）。副本拥有者可以使用 PUT ACL 请求直接更改与副本关联的 ACL，但不能通过复制进行更改。

要指定拥有者覆盖选项，请向每个 Destination 元素添加以下元素：

- AccessControlTranslation 元素，指示 Amazon S3 更改副本所有权
- Account 元素，指定目标存储桶拥有者的 AWS 账户

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    ...
    <Destination>
      ...
      <AccessControlTranslation>
           <Owner>Destination</Owner>
       </AccessControlTranslation>
      <Account>destination-bucket-owner-account-id</Account>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

以下复制配置示例指示 Amazon S3 将键前缀为 Tax 的对象复制到目标存储桶，并更改副本的所有权。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::destination-bucket</Bucket>
         <Account>destination-bucket-owner-account-id</Account>
         <AccessControlTranslation>
```

```
            <Owner>Destination</Owner>
         </AccessControlTranslation>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

向 Amazon S3 授予更改副本所有权的权限

通过在 IAM 角色关联的权限策略中添加 `s3:ObjectOwnerOverrideToBucketOwner` 操作的权限，可向 Amazon S3 授予更改副本所有权的权限。此角色是复制配置中指定的 IAM 角色，允许 Amazon S3 担任该角色并代表您复制对象。

```
...
{
    "Effect":"Allow",
         "Action":[
       "s3:ObjectOwnerOverrideToBucketOwner"
    ],
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

在目标存储桶策略中添加允许更改副本所有权的权限

目标存储桶的拥有者必须向源存储桶的拥有者授予更改副本所有权的权限。目标存储桶的拥有者向源存储桶的拥有者授予 `s3:ObjectOwnerOverrideToBucketOwner` 操作的权限。这允许目标存储桶拥有者接受对象副本的所有权。以下示例存储桶策略语句说明如何执行此操作。

```
...
{
    "Sid":"1",
    "Effect":"Allow",
    "Principal":{"AWS":"source-bucket-account-id"},
    "Action":["s3:ObjectOwnerOverrideToBucketOwner"],
    "Resource":"arn:aws:s3:::destination-bucket/*"
}
...
```

其他注意事项

当您配置所有权覆盖选项时，请注意以下几点：

- 默认情况下，源对象的拥有者也拥有副本。Amazon S3 复制对象版本及其关联的 ACL。

  如果您添加拥有者覆盖，Amazon S3 将仅复制对象版本，而不复制 ACL。此外，Amazon S3 不会复制对源对象 ACL 的任何后续更改。Amazon S3 在副本上设置 ACL，将完全控制权限授予目标存储桶拥有者。

- 当您更新复制配置以启用或禁用拥有者覆盖时，会发生以下情况。

  - 如果向复制配置添加拥有者覆盖选项：

    Amazon S3 复制对象版本时，它会放弃与源对象关联的 ACL。而是改为在副本上设置 ACL，将完全控制权限授予目标存储桶的拥有者。它不会复制对源对象 ACL 的任何后续更改。但是，此 ACL 更改不会应用于设置拥有者覆盖选项之前已复制的对象版本。设置拥有者覆盖之前所复制的源对象 ACL 上的任何更新仍将继续复制（因为对象及其副本继续具有相同的拥有者）。

  - 如果从复制配置中删除拥有者覆盖选项：

    Amazon S3 将源存储桶中出现的新对象及其关联的 ACL 复制到目标存储桶。对于删除拥有者覆盖之前已复制的对象，Amazon S3 不会复制其 ACL，因为 Amazon S3 进行的对象所有权更改仍保持有效。也就是说，对于在您设置了拥有者覆盖的情况下所复制的对象版本上的 ACL，接下来不会进行复制。

如何更改副本拥有者

当复制配置中的#存储桶和##存储桶由不同的 AWS 账户拥有时，您可以指示 Amazon S3 将副本所有权更改为拥有##存储桶的 AWS 账户。此示例介绍如何使用 Amazon S3 控制台和 AWS CLI 更改副本所有权。有关更多信息，请参阅 更改副本拥有者。

> **Note**
>
> 当您使用 S3 副本，且源和目标存储桶由不同的 AWS 账户 拥有时，目标存储桶的存储桶拥有者可以禁用 ACL（通过对象所有权的强制存储桶拥有者设置）以将副本拥有权更改为拥有目标存储桶的 AWS 账户。此设置模拟现有的所有者覆盖行为，而无需 `s3:ObjectOwnerOverrideToBucketOwner` 权限。这意味着，在强制存储桶拥有者设置的情况下已复制到目标存储桶的所有对象都归目标存储桶拥有者所有。有关对象所有权的更多信息，请参阅为您的存储桶控制对象所有权和禁用 ACL。。

有关在跨账户场景中使用 AWS Key Management Service 服务器端加密配置复制的更多信息，请参阅为跨账户方案授予其他权限。

使用 S3 控制台

如需分步指导，请参阅 为同一账户拥有的源存储桶和目标存储桶配置复制。本主题提供了在两个存储桶由相同和不同的 AWS 账户拥有时如何设置复制配置的说明。

使用 AWS CLI

要使用 AWS CLI 更改副本所有权，您需要创建存储桶，对存储桶启用版本控制，创建一个 IAM 角色，该角色向 Amazon S3 赋予复制对象并将复制配置添加到源存储桶的权限。在复制配置中，指示 Amazon S3 更改副本拥有者。此外，您还将测试该设置。

当源存储桶和目标存储桶由不同的 AWS 账户（AWS CLI）拥有时更改副本所有权

1. 在此示例中，您将在两个不同的 AWS 账户 中创建 *source*（源）存储桶和 *destination*（目标）存储桶。使用两个命名配置文件配置 AWS CLI。此示例分别使用名为 `acctA` 和 `acctB` 的配置文件。有关设置凭证配置文件的更多信息，请参阅《AWS Command Line Interface 用户指南》中的命名配置文件。

   > **Important**
   >
   > 用于此练习的配置文件必须具有必要的权限。例如，在复制配置中，指定 Amazon S3 可担任的 IAM 角色。仅当您所使用的配置文件具有 `iam:PassRole` 权限时，您才能执行此操作。如果您使用管理员用户凭证创建命名配置文件，则可执行所有任务。有关更多信息，请参阅 IAM 用户指南中的向用户授予将角色传递给 AWS 服务的权限。

   您将需要确保这些配置文件具有必要的权限。例如，复制配置包含 Amazon S3 可担任的 IAM 角色。仅当您所使用的命名配置文件具有 `iam:PassRole` 权限时，它才能将此类配置附加到存储桶。如果您在创建这些命名配置文件时指定管理员用户凭证，则这些配置文件都将具有权限。有关更多信息，请参阅 IAM 用户指南中的向用户授予将角色传递给 AWS 服务的权限。

2. 创建#存储桶，并启用版本控制。此示例在美国东部（弗吉尼亚州北部）（us-east-1）区域中创建#存储桶。

```
aws s3api create-bucket \
--bucket source \
--region us-east-1 \
```

```
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket source \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. 创建一个##存储桶，并启用版本控制。此示例在美国西部（俄勒冈州）（us-west-2）区域中创建##存储桶。使用不同于用于 *source*（源）存储桶的AWS 账户配置文件。

```
aws s3api create-bucket \
--bucket destination \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctB
```

```
aws s3api put-bucket-versioning \
--bucket destination \
--versioning-configuration Status=Enabled \
--profile acctB
```

4. 您必须添加权限到##存储桶策略以允许更改副本所有权。

   a. 将以下策略保存到 *destination-bucket-policy*.json。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "destination_bucket_policy_sid",
      "Principal": {
        "AWS": "source-bucket-owner-account-id"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:ReplicateTags",
        "s3:GetObjectVersionTagging"
      ],
      "Effect": "Allow",
      "Resource": [
```

```
        "arn:aws:s3:::destination/*"
      ]
    }
  ]
}
```

b. 将以上策略放置到##存储桶：

```
aws s3api put-bucket-policy --region $ {destination_region} --
bucket $ {destination} --policy file://destination_bucket_policy.json
```

5. 创建一个 IAM 角色。您将在稍后添加到#存储桶的复制配置中指定此角色。Amazon S3 担任此角色以代表您复制对象。分两个步骤创建 IAM 角色：

   - 创建角色。
   - 将权限策略附加到角色。

   a. 创建一个 IAM 角色。

      i. 复制以下信任策略，并将其保存到本地计算机上当前目录中一个名为 s3-role-trust-policy.json 的文件。此策略会向 Amazon S3 授予担任该角色的权限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. 运行以下 AWS CLI 命令以创建角色。

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
```

```
--profile acctA
```

b. 将权限策略附加到角色。

i. 复制以下权限策略，并将其保存到本地计算机上当前目录中一个名为 `s3-role-perm-pol-changeowner.json` 的文件。此策略授予对各种 Amazon S3 存储桶和对象操作的权限。在以下步骤中，您将创建一个 IAM 角色并将此策略附加到该角色。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::source/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::source"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ObjectOwnerOverrideToBucketOwner",
            "s3:ReplicateTags",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":"arn:aws:s3:::destination/*"
      }
   ]
```

```
}
```

ii. 要创建策略并将其附加到角色，请运行以下命令。

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-perm-pol-changeowner.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

6. 向源存储桶添加复制配置。

a. AWS CLI 要求以 JSON 形式指定复制配置。将以下 JSON 保存到本地计算机上当前目录中一个名为 `replication.json` 的文件。在配置中，添加 `AccessControlTranslation` 以指示副本所有权的更改。

```
{
   "Role":"IAM-role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
         },
         "Status":"Enabled",
         "Destination":{
            "Bucket":"arn:aws:s3:::destination",
            "Account":"destination-bucket-owner-account-id",
            "AccessControlTranslation":{
               "Owner":"Destination"
            }
         }
      }
   ]
}
```

b. 编辑 JSON，即提供*##*存储桶拥有者账户 ID 和 *IAM-role-ARN* 的值。保存更改。

c. 要向源存储桶添加复制配置，请运行以下命令。提供*#*存储桶名称。

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket source \
--profile acctA
```

7. 在 Amazon S3 控制台中检查副本所有权。

   a. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

   b. 将对象添加到#存储桶。确认 *destination*（目标）存储桶包含对象副本，并且该副本的所有权已更改为拥有 *destination*（目标）存储桶的AWS 账户。

使用 AWS SDK

有关添加复制配置的代码示例，请参阅 使用 AWS SDK。您将需要适当地修改复制配置。有关概念性信息，请参阅 更改副本拥有者。

使用 S3 Replication Time Control（S3 RTC）满足合规性要求

S3 Replication Time Control（S3 RTC）可以帮助您满足数据复制的合规性要求或业务要求，并提供对 Amazon S3 复制时间的可见性。S3 RTC 会在几秒钟内复制您上传到 Amazon S3 的大多数对象，并在 15 分钟内复制 99.99% 的对象。

默认情况下，S3 RTC 包括 S3 复制指标和 Amazon S3 事件通知，您可以使用它们来监控待复制的 S3 API 操作的总数、待复制对象的总大小以及最大复制时间。您可以独立于 S3 RTC 启用复制指标。有关更多信息，请参阅使用复制指标监控进度。此外，S3 RTC 还提供 `OperationMissedThreshold` 和 `OperationReplicatedAfterThreshold` 事件，如果对象复制超过或在 15 分钟阈值后复制，这些事件会通知存储桶拥有者。

借助 S3 RTC，当对象在 15 分钟内未复制（这种情况很罕见）以及当这些对象在 15 分钟阈值之后复制时，Amazon S3 事件可以通知您。Amazon S3 事件可通过 Amazon SQS、Amazon SNS 或 AWS Lambda 获取。有关更多信息，请参阅 the section called “Amazon S3 事件通知”。

主题

- S3 Replication Time Control
- 使用 S3 RTC 的复制指标
- 使用 Amazon S3 事件通知跟踪复制对象
- S3 RTC 的最佳实践和准则

- 启用 S3 Replication Time Control（S3 RTC）

S3 Replication Time Control

您可以基于新的或现有的复制规则开始使用 S3 Replication Time Control（S3 RTC）。您可以选择将复制规则应用于整个 S3 存储桶，或应用于具有特定前缀或标签的 Amazon S3 对象。启用 S3 RTC 时，复制指标也会在复制规则上启用。

如果您使用的是最新版本的复制配置（即在复制配置规则中指定 `Filter` 元素），默认情况下，Amazon S3 不会复制该删除标记。但是，您可以将删除标记复制添加到非基于标记的规则。

> **Note**
>
> 复制指标的计费费率与 Amazon CloudWatch 自定义指标的费率相同。有关信息，请参阅 Amazon CloudWatch 定价。

有关使用 S3 RTC 创建规则的更多信息，请参阅启用 S3 Replication Time Control（S3 RTC）。

使用 S3 RTC 的复制指标

启用了 S3 Replication Time Control（S3 RTC）的复制规则可发布复制指标。对于复制指标，您可以监控等待复制的 S3 API 操作的总数、等待复制的对象的总大小、到目标区域的最长复制时间以及复制失败的操作总数。然后，您可以监控单独复制的每个数据集。

复制指标在启用 S3 RTC 之后的 15 分钟内可用。复制指标可通过 Amazon S3 控制台、Amazon S3 API、AWS SDK、AWS Command Line Interface (AWS CLI) 和 Amazon CloudWatch 提供。有关更多信息，请参阅 使用 Amazon CloudWatch 监控指标。

有关通过 Amazon S3 控制台查找复制指标的更多信息，请参阅使用 Amazon S3 控制台查看复制指标。

使用 Amazon S3 事件通知跟踪复制对象

您可以通过监控 S3 Replication Time Control（S3 RTC）发布的特定事件通知来跟踪 15 分钟内未复制的对象的复制时间。当有资格使用 S3 RTC 进行复制的对象在 15 分钟内未复制以及当这些对象在 15 分钟阈值之后复制时，会发布这些事件。

复制事件在启用 S3 RTC 之后的 15 分钟内可用。Amazon S3 事件可通过 Amazon SQS、Amazon SNS 或 AWS Lambda 获取。有关更多信息，请参阅 Amazon S3 事件通知。

S3 RTC 的最佳实践和准则

在 Amazon S3 中使用 S3 Replication Time Control（S3 RTC）复制数据时，请遵循这些最佳实践准则为您的工作负载优化复制性能。

主题

Amazon S3 复制和请求速率性能准则

当从 Amazon S3 上传和检索存储时，您的应用程序可以实现每秒数千个事务的请求性能。例如，应用程序在 S3 存储桶中每个前缀每秒可实现至少 3,500 个 PUT/COPY/POST/DELETE 请求或 5,500 个 GET/HEAD 请求，包括 S3 复制代表您发出的请求。对存储桶中的前缀数量没有限制。您可以通过并行读取来增加读取或写入性能。例如，如果您在 S3 存储桶中创建 10 个前缀以并行处理读取，则可以将读取性能扩展到每秒 55,000 个读取请求。

Amazon S3 根据高于这些准则的持续请求速率或与 LIST 请求并行的持续请求速率自动扩展。尽管 Amazon S3 在内部针对新的请求速率进行优化，但您可能暂时会收到 HTTP 503 请求响应，直至优化完成。随着每秒请求速率增大，或者当您首次启用 S3 RTC 时，可能会发生这种情况。在这些期间，您的复制延迟可能会增加。S3 RTC 服务等级协议 (SLA) 不适用于超出 Amazon S3 每秒请求性能准则的时间段。

在复制数据传输速率超过默认 1 Gbps 限制的时间段内，S3 RTC SLA 也不适用。如果您预计复制传输速率超过 1 Gbps，则可以联系 AWS Support 中心或使用 Service Quotas 来请求提高限制。

估算您的复制请求速率

您的总请求速率（包括代表您发出的 Amazon S3 复制请求）不得超出复制源存储桶和目标存储桶的 Amazon S3 请求速率准则。对于已复制的每个对象，Amazon S3 复制向源存储桶发出最多五个 GET/HEAD 请求和一个 PUT 请求，向每个目标存储桶发出一个 PUT 请求。

例如，如果您希望每秒复制 100 个对象，则 Amazon S3 复制可能会代表您执行额外 100 个 PUT 请求，每秒总计向源 S3 存储桶发出 200 个 PUT 请求。Amazon S3 复制也可能最多执行 500 个 GET/HEAD 请求（每个已复制的对象有 5 个 GET/HEAD 请求）。

> **Note**
>
> 每个已复制的对象只会产生一个 PUT 请求的费用。有关更多信息，请参阅 Amazon S3 复制常见问题解答中的定价信息。

超过 S3 RTC 数据传输速率限制

如果您预计 S3 Replication Time Control 数据传输速率超过默认的 1Gbps 限制，请联系 AWS Support 中心或使用服务限额请求提高限制。

AWS KMS 加密对象复制请求速率

当您使用 Amazon S3 复制来复制通过服务器端加密 (SSE-KMS) 加密的对象时，AWS Key Management Service (AWS KMS) 每秒请求限制适用。因为您的请求速率超过了每秒请求数的限制，AWS KMS 可能会拒绝其他有效的请求。如果请求受到限制，AWS KMS 将返回 `ThrottlingException` 错误。AWS KMS 请求速率限制适用于您直接发出的请求和代表您通过 Amazon S3 复制发出的请求。

例如，如果您希望每秒复制 1,000 个对象，则可以从 AWS KMS 请求速率限制中减去 2,000 个请求。生成的每秒请求速率可用于除复制之外的 AWS KMS 工作负载。您可以使用 Amazon CloudWatch 中的 AWS KMS 请求指标来监控AWS 账户的总 AWS KMS 请求速率。

启用 S3 Replication Time Control（S3 RTC）

S3 Replication Time Control（S3 RTC）可以帮助您满足数据复制的合规性要求或业务要求，并提供对 Amazon S3 复制时间的可见性。S3 RTC 会在几秒钟内复制您上传到 Amazon S3 的大多数对象，并在 15 分钟内复制 99.99% 的对象。

借助 S3 RTC，您可以监控等待复制的对象总数和大小，以及到目标区域的最长复制时间。复制指标可通过AWS Management Console和 Amazon CloudWatch 用户指南获得。有关更多信息，请参阅 the section called “CloudWatch 中的 S3 复制指标 ”。

使用 S3 控制台

如需分步指导，请参阅 为同一账户拥有的源存储桶和目标存储桶配置复制。本主题提供了在两个存储桶由相同和不同的 AWS 账户 拥有时在复制配置中启用 S3 RTC 的说明。

使用 AWS CLI

要在启用 S3 RTC 的情况下使用 AWS CLI 复制对象，您需要创建存储桶，对存储桶启用版本控制，创建一个 IAM 角色（该角色向 Amazon S3 赋予复制对象的权限），并将复制配置添加到源存储桶。复制配置需要启用 S3 Replication Time Control（S3 RTC）。

在已启用 S3 RTC 的情况下进行复制 (AWS CLI)

- 以下示例设置 `ReplicationTime` 和 `Metric`，并将复制配置添加到源存储桶。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Disabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::destination",
                "Metrics": {
                    "Status": "Enabled",
                    "EventThreshold": {
                        "Minutes": 15
                    }
                },
                "ReplicationTime": {
                    "Status": "Enabled",
                    "Time": {
                        "Minutes": 15
                    }
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

> **Important**
>
> `Metrics:EventThreshold:Minutes` 和 `ReplicationTime:Time:Minutes` 只能设置为 15 作为有效值。

使用适用于 Java 的 AWS SDK

下面是使用 S3 Replication Time Control（S3 RTC）添加复制配置的 Java 示例。

```
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.DeleteMarkerReplication;
import software.amazon.awssdk.services.s3.model.Destination;
import software.amazon.awssdk.services.s3.model.Metrics;
import software.amazon.awssdk.services.s3.model.MetricsStatus;
import software.amazon.awssdk.services.s3.model.PutBucketReplicationRequest;
import software.amazon.awssdk.services.s3.model.ReplicationConfiguration;
import software.amazon.awssdk.services.s3.model.ReplicationRule;
import software.amazon.awssdk.services.s3.model.ReplicationRuleFilter;
import software.amazon.awssdk.services.s3.model.ReplicationTime;
import software.amazon.awssdk.services.s3.model.ReplicationTimeStatus;
import software.amazon.awssdk.services.s3.model.ReplicationTimeValue;

public class Main {

  public static void main(String[] args) {
    S3Client s3 = S3Client.builder()
      .region(Region.US_EAST_1)
      .credentialsProvider(() -> AwsBasicCredentials.create(
          "AWS_ACCESS_KEY_ID",
          "AWS_SECRET_ACCESS_KEY")
      )
      .build();

    ReplicationConfiguration replicationConfig = ReplicationConfiguration
      .builder()
      .rules(
          ReplicationRule
            .builder()
            .status("Enabled")
            .priority(1)
```

```
            .deleteMarkerReplication(
                DeleteMarkerReplication
                    .builder()
                    .status("Disabled")
                    .build()
            )
            .destination(
                Destination
                    .builder()
                    .bucket("destination_bucket_arn")
                    .replicationTime(
                        ReplicationTime.builder().time(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            ReplicationTimeStatus.ENABLED
                        ).build()
                    )
                    .metrics(
                        Metrics.builder().eventThreshold(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            MetricsStatus.ENABLED
                        ).build()
                    )
                    .build()
            )
            .filter(
                ReplicationRuleFilter
                    .builder()
                    .prefix("testtest")
                    .build()
            )
        .build())
        .role("role_arn")
        .build();

   // Put replication configuration
   PutBucketReplicationRequest putBucketReplicationRequest =
PutBucketReplicationRequest
     .builder()
     .bucket("source_bucket")
     .replicationConfiguration(replicationConfig)
     .build();
```

```
    s3.putBucketReplication(putBucketReplicationRequest);
  }
}
```

有关更多信息，请参阅 使用 S3 Replication Time Control（S3 RTC）满足合规性要求。

复制加密对象（SSE-C、SSE-S3、SSE-KMS、DSSE-KMS）

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

在复制已使用服务器端加密进行加密的对象时，有一些特殊注意事项。Amazon S3 支持以下服务器端加密类型：

- 具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）
- 具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）
- 具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）
- 具有客户提供密钥的服务器端加密（SSE-C）

有关服务器端加密的更多信息，请参阅 the section called “服务器端加密”。

本主题介绍指示 Amazon S3 复制已使用服务器端加密进行加密的对象所需的权限。本主题还提供了您可以添加的其他配置元素以及授予复制加密对象所需权限的示例 AWS Identity and Access Management（IAM）策略。

有关带分步说明的示例，请参阅为加密对象启用复制。有关创建复制配置的信息，请参阅复制对象概述。

> **Note**
>
> 您可以在 Amazon S3 中使用多区域 AWS KMS keys。但是，Amazon S3 目前将多区域密钥视为单区域密钥，且不使用密钥的多区域特征。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的使用多区域密钥。

主题

默认存储桶加密如何影响复制

在为复制目标存储桶启用默认加密后，将应用以下加密行为：

- 如果未对源存储桶中的对象进行加密，则将使用目标存储桶的默认加密设置对目标存储桶中的副本对象进行加密。因此，源对象的实体标签（ETag）与副本对象的 ETag 不同。如果您有使用 ETag 的应用程序，则必须更新这些应用程序以弥补这种差异。
- 如果使用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）、具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）或具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）来加密源存储桶中的对象，则目标存储桶中的副本对象使用与源对象相同类型的加密。不会使用目标存储桶的默认加密设置。

复制使用 SSE-C 加密的对象

使用具有客户提供密钥的服务器端加密（SSE-C），您可以管理您自己的专有加密密钥。使用 SSE-C，您可以管理密钥，而 Amazon S3 管理加密和解密过程。您必须在请求中提供加密密钥，但无需编写任何代码即可执行对象加密或解密。在上传对象时，Amazon S3 将使用您提供的密钥加密对象。然后，Amazon S3 将该密钥从内存中清除。在检索对象时，必须提供相同的加密密钥作为您请求的一部分。有关更多信息，请参阅 the section called “客户提供的加密密钥（SSE-C）”。

S3 复制支持使用 SSE-C 加密的对象。您可以在 Amazon S3 控制台中或使用 AWS SDK 配置 SSE-C 对象复制，方法与为未加密对象配置复制的方法相同。除了复制当前所需的权限外，没有其他的 SSE-C 权限。

根据在 S3 复制配置中指定的内容，如果符合条件，S3 复制会自动复制新上传的 SSE-C 加密对象。要复制存储桶中的现有对象，请使用 S3 批量复制。有关复制对象的更多信息，请参阅the section called “设置实时复制”和the section called “复制现有对象”。

复制 SSE-C 对象不会产生额外的费用。有关复制定价的详细信息，请参阅 Amazon S3 定价页面。

复制使用 SSE-S3、SSE-KMS 或 DSSE-KMS 加密的对象

默认情况下，Amazon S3 不会复制使用 SSE-KMS 或 DSSE-KMS 加密的对象。本节介绍您可以添加的其他配置元素以指示 Amazon S3 复制这些对象。

有关带分步说明的示例，请参阅为加密对象启用复制。有关创建复制配置的信息，请参阅复制对象概述。

在复制配置中指定其他信息

在复制配置中，您需要执行以下操作：

- 在复制配置的 `Destination` 元素中，添加您希望 Amazon S3 用来加密对象副本的对称 AWS KMS 客户托管密钥的 ID，如下面的复制配置示例所示。
- 明确选择支持复制使用 KMS 密钥（SSE-KMS 或 DSSE-KMS）加密的对象。要选择加入，请添加 `SourceSelectionCriteria` 元素，如以下复制配置示例所示。

```
<ReplicationConfiguration>
   <Rule>
      ...
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
           <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>

      <Destination>
          ...
          <EncryptionConfiguration>
             <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS #
# as the destination bucket.</ReplicaKmsKeyID>
          </EncryptionConfiguration>
       </Destination>
      ...
   </Rule>
```

```
</ReplicationConfiguration>
```

> **Important**
>
> 必须事先已在目标存储桶所在的同一 AWS 区域 中创建 KMS 密钥。
> KMS 密钥必须有效。PutBucketReplication API 操作不检查 KMS 密钥的有效性。如果使用的 KMS 密钥无效，您将在响应中收到 HTTP 200 OK 状态代码，但复制将失败。

以下示例显示了一个包含可选配置元素的复制配置。此复制配置包含一个规则。该规则应用于键前缀为 Tax 的对象。Amazon S3 使用指定的 AWS KMS key ID 来加密这些对象副本。

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration>
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
         <EncryptionConfiguration>
            <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same
 AWS ## as the destination bucket. (S3 uses this key to encrypt object replicas.)</
ReplicaKmsKeyID>
         </EncryptionConfiguration>
      </Destination>
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
            <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>
   </Rule>
</ReplicationConfiguration>
```

为 IAM 角色授予额外权限

要复制使用 SSE-S3、SSE-KMS 或 DSSE-KMS 静态加密的对象，请向您在复制配置中指定的 AWS Identity and Access Management（IAM）角色授予以下其他权限。通过更新与 IAM 角色关联的权限策略，可授予这些权限。

- 针对源对象的 **`s3:GetObjectVersionForReplication`** 操作 – 此操作允许 Amazon S3 复制未加密的对象，以及通过使用 SSE-S3、SSE-KMS 或 DSSE-KMS 的服务器端加密创建的对象。

  > **Note**
  >
  > 我们建议您使用 `s3:GetObjectVersionForReplication` 操作而不是 `s3:GetObjectVersion` 操作，因为 `s3:GetObjectVersionForReplication` 仅向 Amazon S3 提供进行复制所需的最低权限。此外，`s3:GetObjectVersion` 操作允许复制未加密的对象和 SSE-S3 加密的对象，但不允许复制使用 KMS 密钥（SSE-KMS 或 DSSE-KMS）加密的对象。

- 针对 KMS 密钥的 **`kms:Decrypt`** 和 **`kms:Encrypt`** AWS KMS 操作
  - 您必须授予对 AWS KMS key（用于解密源对象）的 `kms:Decrypt` 权限。
  - 您必须授予对 `kms:Encrypt`（用于加密对象副本）的 AWS KMS key 权限。
- 用于复制明文对象的 **`kms:GenerateDataKey`** 操作 – 如果将明文对象复制到默认情况下启用 SSE-KMS 或 DSSE-KMS 加密的存储桶中，则必须在 IAM 策略中包含目标加密上下文的 `kms:GenerateDataKey` 权限和 KMS 密钥。

我们建议您通过使用 AWS KMS 条件键将这些权限仅限于目标存储桶和对象。对于策略中列出的 KMS 密钥，拥有 IAM 角色的 AWS 账户必须具有执行 `kms:Encrypt` 和 `kms:Decrypt` 操作的权限。如果 KMS 密钥由另一个 AWS 账户拥有，则 KMS 密钥的拥有者必须向拥有 IAM 角色的 AWS 账户授予这些权限。有关管理对这些 KMS 密钥访问的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的将 IAM 策略与 AWS KMS 结合使用。

S3 存储桶密钥和复制

要将复制与 S3 存储桶密钥结合使用，用于加密对象副本的 KMS 密钥的 AWS KMS key 策略必须包含发出调用的主体的 `kms:Decrypt` 权限。对 `kms:Decrypt` 的调用会在使用 S3 存储桶密钥之前验证 S3 存储桶密钥的完整性。有关更多信息，请参阅 将 S3 存储桶密钥与复制功能结合使用。

当为源存储桶或目标存储桶启用 S3 存储桶密钥时，加密上下文将是存储桶的 Amazon 资源名称（ARN），而不是对象的 ARN（例如 `arn:aws:s3:::bucket_ARN`）。您必须更新 IAM 策略才能将存储桶 ARN 用于加密上下文：

```
"kms:EncryptionContext:aws:s3:arn": [
"arn:aws:s3:::bucket_ARN"
]
```

有关更多信息，请参阅加密上下文（x-amz-server-side-encryption-context）（在“使用 REST API”一节中）和启用 S3 存储桶密钥之前需要注意的更改。

示例策略 – 将 SSE-S3 和 SSE-KMS 与复制结合使用

以下 IAM 策略示例显示了将 SSE-S3 和 SSE-KMS 与复制结合使用的语句。

Example – 将 SSE-KMS 用于单独的目标存储桶

以下示例策略显示了将 SSE-KMS 与单独的目标存储桶结合使用的语句。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action": ["kms:Decrypt"],
         "Effect": "Allow",
         "Condition": {
             "StringLike": {
                  "kms:ViaService": "s3.source-bucket-region.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn": [
                      "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
                  ]
             }
          },
          "Resource": [
             "List of AWS KMS key ARNs that are used to encrypt source objects."
          ]
      },
      {
         "Action": ["kms:Encrypt"],
         "Effect": "Allow",
         "Condition": {
            "StringLike": {
               "kms:ViaService": "s3.destination-bucket-1-region.amazonaws.com",
```

```
                "kms:EncryptionContext:aws:s3:arn": [
                   "arn:aws:s3:::amzn-s3-demo-destination-bucket1/key-prefix1*"
                ]
          }
          },
          "Resource": [
             "AWS KMS key ARNs (in the same AWS ## as destination bucket 1). Used to
 encrypt object replicas created in destination bucket 1."
          ]
       },
       {
          "Action": ["kms:Encrypt"],
          "Effect": "Allow",
          "Condition": {
             "StringLike": {
                "kms:ViaService": "s3.destination-bucket-2-region.amazonaws.com",
                "kms:EncryptionContext:aws:s3:arn": [
                   "arn:aws:s3:::amzn-s3-demo-destination-bucket2/key-prefix1*"
                ]
          }
          },
          "Resource": [
             "AWS KMS key ARNs (in the same AWS ## as destination bucket 2). Used to
 encrypt object replicas created in destination bucket 2."
          ]
       }
    ]
}
```

Example – 复制使用 SSE-S3 和 SSE-KMS 创建的对象

以下是一个完整的 IAM 策略，它授予所需的权限以复制未加密的对象、使用 SSE-S3 创建的对象以及使用 SSE-KMS 创建的对象。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
```

```
            "Resource":[
               "arn:aws:s3:::amzn-s3-demo-source-bucket"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:GetObjectVersionForReplication",
               "s3:GetObjectVersionAcl"
            ],
            "Resource":[
               "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:ReplicateObject",
               "s3:ReplicateDelete"
            ],
            "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/key-prefix1*"
         },
         {
            "Action":[
               "kms:Decrypt"
            ],
            "Effect":"Allow",
            "Condition":{
               "StringLike":{
                  "kms:ViaService":"s3.source-bucket-region.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn":[
                     "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
                  ]
               }
            },
            "Resource":[
              "List of the AWS KMS key ARNs that are used to encrypt source objects."
            ]
         },
         {
            "Action":[
               "kms:Encrypt"
            ],
            "Effect":"Allow",
```

```
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.destination-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::amzn-s3-demo-destination-bucket/prefix1*"
               ]
            }
         },
         "Resource":[
            "AWS KMS key ARNs (in the same AWS ## as the destination bucket) to use for
 encrypting object replicas"
         ]
      }
   ]
}
```

Example – 使用 S3 存储桶密钥复制对象

以下是一个完整的 IAM 策略，它授予复制具有 S3 存储桶密钥的对象所必需的权限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
         ]
      },
      {
```

```
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/key-prefix1*"
      },
      {
         "Action":[
            "kms:Decrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.source-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::amzn-s3-demo-source-bucket"
               ]
            }
         },
         "Resource":[
           "List of the AWS KMS key ARNs that are used to encrypt source objects."
         ]
      },
      {
         "Action":[
            "kms:Encrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.destination-bucket-region.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::amzn-s3-demo-destination-bucket"
               ]
            }
         },
         "Resource":[
            "AWS KMS key ARNs (in the same AWS ## as the destination bucket) to use for
encrypting object replicas"
         ]
      }
   ]
```

```
}
```

为跨账户方案授予其他权限

在跨账户方案中（其中源存储桶和目标存储桶由不同的 AWS 账户拥有），您可以使用 KMS 密钥加密对象副本。但是，KMS 密钥拥有者必须向源存储桶拥有者授予使用 KMS 密钥的权限。

> **Note**
>
> 如果您需要跨账户复制 SSE-KMS 数据，则复制规则必须指定来自目标账户 AWS KMS 的客户托管式密钥。AWS 托管式密钥不支持跨账户使用，因此不能用于执行跨账户复制。

向源存储桶拥有者授予使用 KMS 密钥的权限（AWS KMS 控制台）

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS KMS 控制台：https://console.aws.amazon.com/kms。
2. 要更改 AWS 区域，请使用页面右上角的区域选择器。
3. 要查看您账户中自己所创建和管理的密钥，请在导航窗格中选择 Customer managed keys (客户托管密钥)。
4. 请选择 KMS 密钥。
5. 在一般配置部分下，选择密钥策略选项卡。
6. 向下滚动到其他 AWS 账户。
7. 选择添加其他 AWS 账户。

   将显示其他 AWS 账户对话框。
8. 在此对话框中，选择添加其他 AWS 账户。对于 arn:aws:iam::，输入源存储桶账户 ID。
9. 选择 Save changes（保存更改）。

向源存储桶拥有者授予使用 KMS 密钥的权限 (AWS CLI)

- 有关 `put-key-policy` AWS Command Line Interface（AWS CLI）命令的信息，请参阅《AWS CLI 命令参考》中的 put-key-policy。有关基础 `PutKeyPolicy` API 操作的信息，请参阅《AWS Key Management Service API 参考》https://docs.aws.amazon.com/kms/latest/APIReference/中的 PutKeyPolicy。

AWS KMS 事务限额注意事项

在启用跨区域复制（CRR）后使用 AWS KMS 加密添加许多新对象时，您可能会遇到节流（HTTP `503 Service Unavailable` 错误）。如果每秒 AWS KMS 事务数超出当前限额，则会发生节流。有关更多信息，请参阅《AWS Key Management Service 开发人员指南》中的配额。

要请求增加限额，您可以使用服务限额。有关更多信息，请参阅 Requesting a quota increase（请求增加限额）。如果您所在区域不支持服务配额，请创建一个 AWS Support 案例。

为加密对象启用复制

默认情况下，Amazon S3 不会复制使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）或具有 AWS KMS 密钥的双层服务器端加密（DSSE-KMS）加密的对象。要复制用 SSE-KMS 或 DSS-KMS 加密的对象，必须修改存储桶复制配置，以指示 Amazon S3 复制这些对象。此示例介绍如何使用 Amazon S3 控制台和 AWS Command Line Interface（AWS CLI）更改存储桶复制配置以允许复制加密对象。

有关更多信息，请参阅 复制加密对象（SSE-C、SSE-S3、SSE-KMS、DSSE-KMS）。

> **Note**
>
> 当为源或目标存储桶启用 S3 存储桶密钥时，加密上下文将是存储桶的 Amazon 资源名称 (ARN)，而不是对象的 ARN。您必须更新 IAM 策略才能将存储桶 ARN 用于加密上下文。有关更多信息，请参阅 S3 存储桶密钥和复制。

> **Note**
>
> 您可以在 Amazon S3 中使用多区域 AWS KMS keys。但是，Amazon S3 目前将多区域密钥视为单区域密钥，且不使用密钥的多区域特征。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的使用多区域密钥。

使用 S3 控制台

如需分步指导，请参阅 为同一账户拥有的源存储桶和目标存储桶配置复制。本主题提供了在两个存储桶由相同和不同的 AWS 账户拥有时设置复制配置的说明。

使用 AWS CLI

要使用 AWS CLI 复制加密的对象，请执行以下操作：

- 创建源存储桶和目标存储桶，并对这些存储桶启用版本控制。
- 创建向 Amazon S3 授予复制对象的权限的 AWS Identity and Access Management（IAM）服务角色。IAM 角色的权限包含复制这些加密对象所需的权限。
- 将复制配置添加到源存储桶。复制配置提供与复制使用 KMS 密钥加密的对象相关的信息。
- 将加密的对象添加到源存储桶。
- 测试设置，以确认加密的对象正在复制到目标存储桶。

以下过程指导您完成此过程。

复制服务器端加密对象 (AWS CLI)

1. 在此示例中，我们在同一个 AWS 账户中创建了 *amzn-s3-demo-source-bucket* 存储桶和 *amzn-s3-demo-destination-bucket* 存储桶。还可以为 AWS CLI 设置凭证配置文件。此示例使用配置文件名称 *acctA*。

   有关设置凭证配置文件的更多信息，请参阅《AWS Command Line Interface 用户指南》中的命名配置文件。要使用本示例中的命令，请将 *user input placeholders* 替换为您的信息。

2. 使用以下命令创建 *DOC-EXAMPLE-SOURCE-BUCKET* 存储桶并对该存储桶启用版本控制。以下示例命令在美国东部（弗吉尼亚州北部）（us-east-1）区域中创建 *DOC-EXAMPLE-SOURCE-BUCKET* 存储桶。

   ```
   aws s3api create-bucket \
   --bucket DOC-EXAMPLE-SOURCE-BUCKET \
   --region us-east-1 \
   --profile acctA
   ```

   ```
   aws s3api put-bucket-versioning \
   --bucket DOC-EXAMPLE-SOURCE-BUCKET \
   --versioning-configuration Status=Enabled \
   --profile acctA
   ```

3. 使用以下命令创建 *DOC-EXAMPLE-DESTINATION-BUCKET* 存储桶并对该存储桶启用版本控制。以下示例命令在美国西部（俄勒冈州）（us-west-2）区域中创建 *DOC-EXAMPLE-DESTINATION-BUCKET* 存储桶。

> **Note**
>
> 要在 *DOC-EXAMPLE-SOURCE-BUCKET* 存储桶和 *DOC-EXAMPLE-DESTINATION-BUCKET* 存储桶均位于同一 AWS 账户中时设置复制配置，请使用同一配置文件。在此示例中，我们使用的是 *acctA*。要在两个存储桶由不同 AWS 账户拥有时配置复制，您需要为每个存储桶指定不同的配置文件。

```
aws s3api create-bucket \
--bucket DOC-EXAMPLE-DESTINATION-BUCKET \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket DOC-EXAMPLE-DESTINATION-BUCKET \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. 接下来，创建 IAM 服务角色。您将在稍后添加到 *DOC-EXAMPLE-SOURCE-BUCKET* 存储桶的复制配置中指定此角色。Amazon S3 担任此角色以代表您复制对象。分两个步骤创建 IAM 角色：

   - 创建服务角色。
   - 将权限策略附加到角色。

   a. 要创建 IAM 服务角色，请执行以下操作：

      i. 复制以下信任策略，并将其保存到本地计算机上当前目录中一个名为 `s3-role-trust-policy-kmsobj.json` 的文件。此策略授予 Amazon S3 服务主体代入该角色的权限，以便 Amazon S3 可代表您执行任务。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
```

```
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

ii. 使用以下命令创建角色：

```
$ aws iam create-role \
--role-name replicationRolekmsobj \
--assume-role-policy-document file://s3-role-trust-policy-kmsobj.json  \
--profile acctA
```

b. 接下来，将权限策略附加到角色。此策略授予对各种 Amazon S3 存储桶和对象操作的权限。

i. 复制以下权限策略，并将其保存到本地计算机上当前目录中一个名为 s3-role-permissions-policykmsobj.json 的文件。您将创建一个 IAM 角色，稍后将策略附加到该角色。

> **Important**
>
> 在权限策略中，您可以指定将用于加密 *amzn-s3-demo-source-bucket* 和 *amzn-s3-demo-destination-bucket* 存储桶的 AWS KMS 密钥 ID。您必须为 *amzn-s3-demo-source-bucket* 和 *amzn-s3-demo-destination-bucket* 存储桶创建两个独立的 KMS 密钥。AWS KMS keys不会在创建它们的 AWS 区域之外共享。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Effect":"Allow",
```

```
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket",
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLikeIfExists":{
               "s3:x-amz-server-side-encryption":[
                  "aws:kms",
                  "AES256",
                  "aws:kms:dsse"
               ],
               "s3:x-amz-server-side-encryption-aws-kms-key-id":[
                  "AWS KMS key IDs(in ARN format) to use for encrypting
object replicas"
               ]
            }
         },
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      },
      {
         "Action":[
            "kms:Decrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.us-east-1.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
               ]
            }
         },
         "Resource":[
            "AWS KMS key IDs(in ARN format) used to encrypt source
objects."
         ]
```

```
      },
      {
         "Action":[
            "kms:Encrypt"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLike":{
               "kms:ViaService":"s3.us-west-2.amazonaws.com",
               "kms:EncryptionContext:aws:s3:arn":[
                  "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
               ]
            }
         },
         "Resource":[
            "AWS KMS key IDs(in ARN format) to use for encrypting object
 replicas"
         ]
      }
   ]
}
```

ii. 创建一个策略并将其附加到角色。

```
$ aws iam put-role-policy \
--role-name replicationRolekmsobj \
--policy-document file://s3-role-permissions-policykmsobj.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

5. 接下来，将以下复制配置添加到 *amzn-s3-demo-source-bucket* 存储桶。它指示 Amazon S3 将前缀为 Tax/ 的对象复制到 *amzn-s3-demo-destination-bucket* 存储桶。

> **Important**
>
> 在复制配置中，指定 Amazon S3 可代入的 IAM 角色。仅当您具有 iam:PassRole 权限时，才能执行此操作。您在 CLI 命令中指定的配置文件必须具有此权限。有关更多信息，请参阅《IAM 用户指南》中的向用户授予将角色传递给 AWS 服务的权限。

```
<ReplicationConfiguration>
 <Role>IAM-Role-ARN</Role>
```

```
  <Rule>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>Disabled</Status>
    </DeleteMarkerReplication>
    <Filter>
       <Prefix>Tax</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <SourceSelectionCriteria>
      <SseKmsEncryptedObjects>
        <Status>Enabled</Status>
      </SseKmsEncryptedObjects>
    </SourceSelectionCriteria>
    <Destination>
      <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
      <EncryptionConfiguration>
        <ReplicaKmsKeyID>AWS KMS key IDs to use for encrypting object replicas</
ReplicaKmsKeyID>
      </EncryptionConfiguration>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

要将复制配置添加到 *amzn-s3-demo-source-bucket* 存储桶，请执行以下操作：

a. AWS CLI 要求以 JSON 形式指定复制配置。将以下 JSON 保存到本地计算机上当前目录中的一个文件 (replication.json)。

```
{
   "Role":"IAM-Role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
            "Prefix":"Tax"
         },
         "Destination":{
            "Bucket":"arn:aws:s3:::amzn-s3-demo-destination-bucket",
```

```
            "EncryptionConfiguration":{
               "ReplicaKmsKeyID":"AWS KMS key IDs (in ARN format) to use for
 encrypting object replicas"
            }
         },
         "SourceSelectionCriteria":{
            "SseKmsEncryptedObjects":{
               "Status":"Enabled"
            }
         }
      }
   ]
}
```

b. 编辑 JSON 以提供 *amzn-s3-demo-destination-bucket* 存储桶、*AWS KMS key IDs (in ARN format)* 和 *IAM-role-ARN* 的值。保存更改。

c. 使用以下命令以将复制配置添加到 *amzn-s3-demo-source-bucket* 存储桶中。请务必提供 *amzn-s3-demo-source-bucket* 存储桶名称。

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket amzn-s3-demo-source-bucket \
--profile acctA
```

6. 测试该配置以确认已经复制了加密对象。在 Amazon S3 控制台中，执行以下操作：

a. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

b. 在 *amzn-s3-demo-source-bucket* 存储桶中，创建一个名为 Tax 的文件夹。

c. 将示例对象添加到文件夹。请务必选择加密选项并指定用于加密对象的 KMS 密钥。

d. 确认 *amzn-s3-demo-destination-bucket* 存储桶包含对象副本，并且它们已使用在配置中指定的 KMS 密钥进行了加密。有关更多信息，请参阅 the section called “获取复制状态”。

使用 AWS SDK

有关显示如何添加复制配置的代码示例，请参阅使用 AWS SDK。您必须适当地修改复制配置。

有关概念性信息，请参阅 复制加密对象（SSE-C、SSE-S3、SSE-KMS、DSSE-KMS）。

使用 Amazon S3 副本修改同步复制元数据更改

Amazon S3 副本修改同步可以帮助您在副本和源对象之间保留已复制的对象元数据，例如标签、ACL 和对象锁定设置等等。默认情况下，Amazon S3 仅会从源对象向副本复制元数据。启用副本修改同步后，Amazon S3 会将对副本所做的元数据更改复制回源对象，使复制成为双向复制。

启用副本修改同步

您可以将 Amazon S3 副本修改同步与新的或现有的复制规则一起使用。您可以将其应用于整个 S3 存储桶或具有特定前缀的 Amazon S3 对象。

要使用 Amazon S3 控制台启用副本修改同步，请参阅配置实时复制的示例。本主题提供了在存储桶由同一或不同的 AWS 账户拥有时在复制配置中启用副本修改同步的说明。

要使用 AWS Command Line Interface（AWS CLI）启用副本修改同步，您必须向包含已启用 `ReplicaModifications` 的副本的存储桶添加复制配置。要设置双向复制，请创建从源存储桶（*amzn-s3-demo-bucket1*）到包含副本的存储桶（*amzn-s3-demo-bucket2*）的复制规则。然后，创建从包含副本的存储桶（*amzn-s3-demo-bucket2*）到源存储桶（*amzn-s3-demo-bucket1*）的第二个复制规则。存储桶所在的 AWS 区域可以相同，也可以不同。

> **Note**
>
> 您必须在两个存储桶上启用副本修改同步，才能复制副本元数据更改，例如被复制对象上的对象访问控制列表（ACL）、对象标记或对象锁定设置。与所有复制规则一样，这些规则既可以应用于整个 Amazon S3 存储桶，也可以应用于按前缀或对象标签筛选的 Amazon S3 对象子集。

在以下示例配置中，Amazon S3 将以 *Tax* 为前缀的元数据更改复制到存储桶 *amzn-s3-demo-bucket*，该存储桶包含源对象。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "SourceSelectionCriteria": {
                "ReplicaModifications":{
                    "Status": "Enabled"
```

```
                }
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

有关使用 AWS CLI 创建复制规则的完整说明，请参阅为同一账户拥有的源存储桶和目标存储桶配置复制。

在存储桶之间复制删除标记

默认情况下，当启用 S3 复制且源存储桶中的对象被删除时，Amazon S3 仅在源存储桶中添加删除标记。此操作会防止恶意删除数据。

如果您启用了删除标记复制，则这些标记将复制到目标存储桶，Amazon S3 会当作对象在源存储桶和目标存储桶中都删除了一样。有关删除标记工作原理的更多信息，请参阅使用删除标记。

> **Note**
>
> 基于标签的复制规则不支持删除标记复制。删除标记复制也不符合使用 S3 Replication Time Control 时授予的 15 分钟 SLA。

如果您没在使用最新的复制配置版本，则删除操作将对复制产生不同的影响。有关更多信息，请参阅删除操作对复制操作有何影响。

启用删除标记复制

您可以开始将删除标记复制与新的或现有的复制规则结合使用。您可以将其应用于整个 S3 存储桶或具有特定前缀的 Amazon S3 对象。

> **Note**
>
> 当您启用删除标记复制并且您的存储桶具有 S3 生命周期到期规则时，S3 生命周期到期规则添加的删除标记将不会复制到目标存储桶。

要使用 Amazon S3 控制台启用删除标记复制，请参阅使用 S3 控制台。本主题提供了在存储桶由同一或不同的 AWS 账户拥有时在复制配置中启用删除标记复制的说明。

要使用 AWS Command Line Interface（AWS CLI）启用删除标记复制，您必须向已启用 DeleteMarkerReplication 的源存储桶添加复制配置。

在以下示例配置中，删除标记将复制到目标存储桶 *DOC-EXAMPLE-BUCKET* 中以 *Tax* 为前缀的对象。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Enabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

有关通过 AWS CLI 创建复制规则的完整说明，请参阅复制演练部分中的为同一账户拥有的源存储桶和目标存储桶配置复制。

管理或暂停实时复制

实时复制是跨相同或不同 AWS 区域中的存储桶自动、异步复制对象。在您设置复制配置后，Amazon S3 会将源存储桶中新创建的对象和对象更新复制到一个或多个指定的目标存储桶。

您使用 Amazon S3 控制台向源存储桶添加复制规则。复制规则定义了要复制的源存储桶对象和存储已复制对象的目标存储桶。有关复制的更多信息，请参阅 复制对象概述。

您可以在 Replication 页面上管理复制规则。可以添加、查看、启用、禁用或删除复制规则。也可以更改复制规则的优先级。有关向存储桶添加复制规则的信息，请参阅 使用 S3 控制台。

使用 Amazon S3 控制台管理 S3 存储桶的复制规则

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在通用存储桶选项卡上，选择您想要的存储桶的名称。
4. 选择管理选项卡，然后向下滚动到复制规则。
5. 可以通过以下方式更改复制规则：
   - 要启用或禁用复制规则，请选择规则左侧的选项按钮。在操作菜单上，选择启用规则或禁用规则。还可从操作菜单中禁用、启用或删除存储桶中的所有规则。

     > **Note**
     >
     > 如果您禁用复制规则，之后又重新启用该规则，那么在禁用规则期间未复制的任何新对象或更改的对象在重新启用该规则后均不会 自动复制。要复制这些对象，必须使用 S3 批量复制。有关更多信息，请参阅 the section called “复制现有对象”。

   - 要更改规则的优先级，请选择规则左侧的选项按钮，然后选择编辑规则。

     您需要设置规则优先级，以避免因在多个规则的范围内包含对象而引起冲突。如果规则重叠，Amazon S3 会使用规则优先级确定要应用哪个规则。数字越大，优先级越高。有关规则优先级的更多信息，请参阅 复制配置。

暂停或停止复制

要临时暂停复制并在以后自动恢复，您可以使用 AWS Fault Injection Service 中的 `aws:s3:bucket-pause-replication` 操作。有关更多信息，请参阅《AWS Fault Injection Service 用户指南》中的 aws:s3:bucket-pause-replication 和 Pause S3 Replication。

要在 Amazon S3 中停止复制，我们建议您禁用复制规则。如果您禁用复制规则，之后又重新启用该规则，那么在禁用规则期间未复制的任何新对象或更改的对象在重新启用该规则后均不会 自动复制。要复制这些对象，必须使用 S3 批量复制。有关更多信息，请参阅 the section called “复制现有对象”。

如果您移除向 Amazon S3 授予所需权限的 AWS Identity and Access Management（IAM）角色、AWS Key Management Service（AWS KMS）权限或存储桶策略权限，复制也将停止。但是，我们建议您不要使用这些方法，因为它们会导致复制失败。Amazon S3 会将受影响对象的复制状态报告

为 FAILED。如果稍后权限恢复，则标记为 FAILED 的对象不会 自动复制。要复制这些对象，必须使用 S3 批量复制。

使用复制指标和 S3 事件通知监控进度

S3 复制指标为复制配置中的复制规则提供了详细的指标。使用复制指标，您可以通过跟踪待复制的字节数、待复制的操作数、复制失败的操作数和复制延迟来逐分钟监控进度。

启用 S3 Replication Time Control（S3 RTC）时，将自动开启 S3 复制指标。在创建或编辑规则时，您还可以独立于 S3 RTC 启用 S3 复制指标。S3 RTC 包含其他功能，例如服务水平协议（SLA）和错过阈值的通知。有关更多信息，请参阅 使用 S3 Replication Time Control（S3 RTC）满足合规性要求。

待复制字节数、待复制操作数和复制延迟指标仅适用于通过 S3 Cross-Region Replication（S3 CRR）或 S3 Same-Region Replication（S3 SRR）复制的新对象。复制失败的操作数指标既跟踪使用 S3 CRR 或 S3 SRR 复制的新对象，也跟踪通过 S3 批量复制所复制的现有对象。为帮助排查任何配置问题，您还可以将 Amazon S3 事件通知设置为接收复制失败事件。

启用后，S3 复制指标会将以下指标发布到 Amazon CloudWatch：

- 待复制的字节数 – 给定复制规则的待复制对象的总字节数。
- 复制延迟 – 对于给定复制规则，复制目标存储桶落后于源存储桶的最大秒数。
- 待复制的操作 – 给定复制规则的等待复制的操作的数量。此指标跟踪与对象、删除标记、标签、访问控制列表（ACL）和 S3 对象锁定相关的操作。
- 复制失败的操作 – 给定复制规则的复制失败的操作数。此指标跟踪与对象、删除标记、标签、ACL 和对象锁定相关的操作。与其他复制指标不同，此指标既适用于使用 S3 CRR 或 S3 SRR 复制的新对象，也适用于使用 S3 批量复制所复制的现有对象。

> **Note**
>
> 复制失败的操作跟踪以一分钟为间隔汇总的 S3 复制失败。要确定复制失败的特定对象及其失败原因，请在 Amazon S3 事件通知中订阅 OperationFailedReplication 事件。有关更多信息，请参阅 使用 Amazon S3 事件通知接收复制失败事件。

如果任务根本无法运行，则指标不会发送到 Amazon CloudWatch。例如，如果您没有运行 S3 批量复制任务所需的权限，或者复制配置中的标签或前缀不匹配，则您的任务将无法运行。

主题

启用 S3 复制指标

您可以开始将 S3 复制指标与新的或现有的复制规则结合使用。您可以选择将复制规则应用于整个 S3 存储桶，或应用于具有特定前缀或标签的 Amazon S3 对象。

本主题提供了在源和目标存储桶由相同或不同的 AWS 账户拥有时，在复制配置中启用 S3 复制指标的说明。

要使用 AWS Command Line Interface（AWS CLI）启用复制指标，您必须向启用 `Metrics` 的源存储桶添加复制配置。在此示例配置中，以 *`Tax`* 为前缀的对象将复制到目标存储桶 *`DOC-EXAMPLE-BUCKET`*，并为这些对象生成指标。

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "Metrics": {
                    "Status": "Enabled"
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

有关创建复制规则的完整说明，请参阅 为同一账户拥有的源存储桶和目标存储桶配置复制。

有关在 S3 控制台中查看复制指标的更多信息，请参阅 使用 Amazon S3 控制台查看复制指标。

> **Note**
>
> S3 复制指标的费率与 Amazon CloudWatch 自定义指标费率相同。有关更多信息，请参阅 Amazon CloudWatch 定价。

使用 Amazon S3 事件通知接收复制失败事件

在对象未复制到其目标 AWS 区域的情况下，S3 事件通知可以通知您。Amazon S3 事件可以通过 Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS) 或 AWS Lambda 获得。有关更多信息，请参阅 the section called “Amazon S3 事件通知”。

有关 S3 事件通知捕获的故障代码的列表，请参阅Amazon S3 复制失败原因。

使用 S3 Storage Lens 存储统计管理工具查看复制指标

要获取 S3 复制的详细指标，包括复制规则计数指标，您可以使用 Amazon S3 Storage Lens 存储统计管理工具。S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。有关更多信息，请参阅使用 S3 Storage Lens 存储统计管理工具保护您的数据。有关指标的完整列表，请参阅 S3 Storage Lens 存储统计管理工具指标词汇表。

使用 Amazon S3 控制台查看复制指标

Amazon S3 存在三种类型的 Amazon CloudWatch 指标：存储指标、请求指标和复制指标。当您通过 AWS Management Console或 Amazon S3 API 使用 S3 Replication Time Control（S3 RTC）启用复制时，S3 复制指标将自动开启。在创建或编辑规则时，您还可以独立于 S3 RTC 启用 S3 复制指标。

复制指标跟踪复制配置的规则 ID。复制规则 ID 可以特定于前缀、标签或两者的组合。

有关 Amazon S3 的 CloudWatch 指标的更多信息，请参阅 使用 Amazon CloudWatch 监控指标。

先决条件

启用具有 S3 复制指标的复制规则。

查看复制指标

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。在存储桶列表中，请选择您要获取其复制指标的对象所在的存储桶的名称。
3. 请选择 Metrics（指标）选项卡。
4. 在 Replication metrics (复制指标) 下，请选择 Replication rules (复制规则)。
5. 请选择 Display charts (显示图表)。

   Amazon S3 显示复制延迟（以秒为单位）、待复制的字节数、待复制的操作和复制失败的操作图表。

然后，您可以查看所选规则的复制指标复制延迟（以秒为单位）、待复制的操作、待复制的字节数和复制失败的操作。如果您使用 S3 Replication Time Control，Amazon CloudWatch 在相应的复制规则上启用 S3 RTC 后 15 分钟开始报告复制指标。您可以在 Amazon S3 控制台或 CloudWatch 控制台上查看复制指标。有关更多信息，请参阅 使用 S3 RTC 的复制指标。

> **Note**
>
> 还可以使用 Amazon S3 Storage Lens 存储统计管理工具在 Amazon S3 控制台中查看 S3 复制的详细指标。S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。有关更多信息，请参阅使用 S3 Storage Lens 存储统计管理工具保护您的数据。有关指标的完整列表，请参阅 S3 Storage Lens 存储统计管理工具指标词汇表。

Amazon S3 复制失败原因

下表列出了 Amazon S3 复制失败原因。您可以通过 Amazon S3 事件通知接收 failureReason 事件来查看这些原因。您可以通过 Amazon Simple Queue Service（Amazon SQS）、Amazon Simple Notification Service（Amazon SNS）或 AWS Lambda 接收 S3 事件通知。有关更多信息，请参阅 Amazon S3 事件通知。

还可以在 S3 批量复制完成报告中查看这些失败原因。有关更多信息，请参阅 分批复制完成报告。

复制失败原因	描述
AssumeRoleNotPermitted	Amazon S3 无法代入在复制配置或批量操作任务中指定的 AWS Identity and Access Management（IAM）角色。
DstBucketInvalidRegion	目标存储桶未处于由批量操作任务指定的同一个 AWS 区域中。此错误特定于分批复制。
DstBucketNotFound	Amazon S3 找不到在复制配置中指定的目标存储桶。
DstBucketObjectLockConfigMissing	要在启用对象锁定的情况下从源存储桶复制对象，目标存储桶也必须启用对象锁定。此错误表示目标存储桶中可能未启用对象锁定。有关更多信息，请参阅 对象锁定注意事项。
DstBucketUnversioned	未为 S3 目标存储桶启用版本控制。要使用 S3 复制来复制对象，请对目标存储桶启用版本控制。
DstDelObjNotPermitted	Amazon S3 无法将删除标记复制到目标存储桶。针对目标存储桶的 s3:ReplicateDelete 权限可能缺失。
DstKmsKeyInvalidState	适用于目标存储桶的 AWS Key Management Service（AWS KMS）密钥处于无效状态。查看并启用所需的 AWS KMS 密钥。有关管理 AWS KMS 密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS 密钥的密钥状态。
DstKmsKeyNotFound	在复制配置中为目标存储桶配置的 AWS KMS 密钥不存在。
DstMultipartCompleteNotPermitted	Amazon S3 无法完成目标存储桶中对象的分段上传。针对目标存储桶的 s3:ReplicateObject 权限可能缺失。

复制失败原因	描述
DstMultipartInitNotPermitted	Amazon S3 无法启动将对象分段上传到目标存储桶。针对目标存储桶的 s3:ReplicateObject 权限可能缺失。
DstMultipartPartUploadNotPermitted	Amazon S3 无法将多分段对象上传到目标存储桶。针对目标存储桶的 s3:ReplicateObject 权限可能缺失。
DstObjectHardDeleted	S3 分批复制不支持重新复制从目标存储桶中使用对象的版本 ID 删除的对象。此错误特定于分批复制。
DstPutAclNotPermitted	Amazon S3 无法将对象访问控制列表（ACL）复制到目标存储桶。针对目标存储桶的 s3:ReplicateObject 权限可能缺失。
DstPutLegalHoldNotPermitted	在复制不可变对象时，Amazon S3 无法对目标对象设置对象锁定法定保留。针对目标存储桶的 s3:PutObjectLegalHold 权限可能缺失。有关更多信息，请参阅 依法保留。
DstPutObjectNotPermitted	Amazon S3 无法将对象复制到目标存储桶。针对目标存储桶的 s3:ReplicateObject 或 s3:ObjectOwnerOverrideToBucketOwner 权限可能缺失。
DstPutTaggingNotPermitted	Amazon S3 无法将对象标签复制到目标存储桶。针对目标存储桶的 s3:ReplicateObject 权限可能缺失。
DstVersionNotFound	Amazon S3 无法在目标存储桶中找到需要复制其元数据的所需对象版本。
InitiateReplicationNotPermitted	Amazon S3 无法在对象上启动复制。批量操作任务可能缺少 s3:InitiateReplication 权限。此错误特定于分批复制。

复制失败原因	描述
SrcBucketInvalidRegion	源存储桶未处于批量操作任务指定的同一个 AWS 区域中。此错误特定于分批复制。
SrcBucketNotFound	Amazon S3 找不到源存储桶。
SrcBucketReplicationConfigMissing	Amazon S3 找不到源存储桶的复制配置。
SrcGetAclNotPermitted	Amazon S3 无法访问源存储桶中要复制的对象。针对源存储桶对象的 s3:GetObjectVersionAcl 权限可能缺失。 源存储桶中的对象必须由存储桶拥有者所有。如果启用了 ACL，则验证应将“对象所有权”设置为“首选存储桶拥有者”还是“对象编写者”。如果将“对象所有权”设置为“首选存储桶拥有者”，则源存储桶对象必须具有 bucket-owner-full-control ACL，存储桶拥有者才能成为对象拥有者。源账户可以通过将“对象所有权”设置为“强制存储桶拥有者”并禁用 ACL，从而获取其存储桶中所有对象的所有权。
SrcGetLegalHoldNotPermitted	Amazon S3 无法访问 S3 对象锁定法定保留信息。
SrcGetObjectNotPermitted	Amazon S3 无法访问源存储桶中要复制的对象。针对源存储桶的 s3:GetObjectVersionForReplication 权限可能缺失。
SrcGetRetentionNotPermitted	Amazon S3 无法访问 S3 对象锁定保留期信息。
SrcGetTaggingNotPermitted	Amazon S3 无法从源存储桶访问对象标签信息。针对源存储桶的 s3:GetObjectVersionTagging 权限可能缺失。

复制失败原因	描述
SrcHeadObjectNotPermitted	Amazon S3 无法从源存储桶检索对象元数据。针对源存储桶的 s3:GetObjectVersionForReplication 权限可能缺失。
SrcKeyNotFound	Amazon S3 找不到要复制的源对象密钥。源对象可能在复制完成之前已被删除。
SrcKmsKeyInvalidState	适用于源存储桶的 AWS KMS 密钥处于无效状态。查看并启用所需的 AWS KMS 密钥。有关管理 AWS KMS 密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS 密钥的密钥状态。
SrcObjectNotEligible	某些对象不符合复制条件。这可能是由于对象的存储类或对象标签与复制配置不匹配所致。
SrcObjectNotFound	源对象不存在。
SrcReplicationNotPending	Amazon S3 已经复制了此对象。此对象不再处于待复制状态。
SrcVersionNotFound	Amazon S3 找不到要复制的源对象版本。源对象版本可能在复制完成之前已被删除。

相关 主题

为实时复制设置权限

对复制进行问题排查

获取复制状态信息

复制状态可以帮助您确定正在复制的对象的当前状态。源对象的复制状态将返回 PENDING、COMPLETED、或 FAILED。副本的复制状态将返回 REPLICA。

主题

复制状态概述

在复制中，您有一个源存储桶（对它配置复制）和一个目标存储桶（Amazon S3 将对象复制到其中）。当您请求这些存储桶中的对象（使用 GET 对象）或对象元数据（使用 HEAD 对象）时，Amazon S3 将在响应中返回 `x-amz-replication-status` 标头：

- 如果请求源存储桶中的对象，Amazon S3 将返回 `x-amz-replication-status` 标头（如果请求中的对象符合复制条件）。

  例如，假设您在复制配置中指定了对象前缀 `TaxDocs`，以指示 Amazon S3 仅复制键名前缀为 `TaxDocs` 的对象。您上传的具有此键名前缀的任何对象（例如 `TaxDocs/document1.pdf`）都将复制。对于具有此键名前缀的对象请求，Amazon S3 会返回其对象复制状态为以下值之一的 `x-amz-replication-status` 标头：`PENDING`、`COMPLETED` 或 `FAILED`。

  > **Note**
  >
  > 如果在上传对象后对象复制失败，您无法重试复制。您必须重新上传对象。对象会因缺少复制角色权限、AWS KMS 权限或存储桶权限等问题而导致转换为 `FAILED` 状态。对于临时故障（例如，在存储桶或区域不可用的情况下），复制状态将不会转换为 `FAILED`，而仍将保持 `PENDING`。在资源重新联机后，S3 将继续复制这些对象。

- 当请求目标存储桶中的对象时，如果请求中的对象是 Amazon S3 创建的副本，Amazon S3 会返回值为 `REPLICA` 的 `x-amz-replication-status` 标头。

> **Note**
>
> 在从启用了复制的源存储桶中删除对象之前，请检查该对象的复制状态，以确保已复制了该对象。
>
> 如果对源存储桶启用了生命周期配置，则 Amazon S3 将暂停生命周期操作，直到它将对象状态标记为 `COMPLETED` 或 `FAILED`。

复制到多个目标存储桶时的复制状态

将对象复制到多个目标存储桶时，`x-amz-replication-status` 标头的作用有所不同。当向所有目标的复制都成功时，源对象的标头仅返回 `COMPLETED` 值。在完成所有目标的复制之前，标头将保持 `PENDING` 值。如果一个或多个目标复制失败，则标头返回 `FAILED`。

启用了 Amazon S3 副本修改同步时的复制状态

当您的复制规则启用了 Amazon S3 副本修改同步时，副本可能报告 `REPLICA` 以外的状态。如果元数据更改正在复制过程中，`x-amz-replication-status` 标头返回 `PENDING`。如果副本修改同步无法复制元数据，则标头返回 `FAILED`。如果正确复制元数据，则副本将返回标头 `REPLICA`。

查找复制状态

要获取存储桶中对象的复制状态，可以使用 Amazon S3 清单工具。Amazon S3 会向目标存储桶发送一个您在清单配置中指定的 CSV 文件。您还可以使用 Amazon Athena 在清单报告中查询复制状态。有关 Amazon S3 清单的更多信息，请参阅 Amazon S3 清单。

您可以使用控制台、AWS Command Line Interface (AWS CLI) 或 AWS SDK 查找对象复制状态。

使用 S3 控制台

在 S3 控制台中，您可以在 Object management overview（对象管理概览）下的对象 Details（详细信息）页面查看对象的复制状态。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择存储桶名称。
3. 在 Objects (对象) 列表中，请选择对象名称。
4. 在 Properties（属性）选项卡下，找到 Object management overview（对象管理概述），您可以在此查看 Replication status（复制状态）。

使用 AWS CLI

使用 `head-object` 命令检索对象元数据，如下所示。

```
aws s3api head-object --bucket source-bucket --key object-key --version-id object-version-id
```

该命令会返回对象元数据（包括 ReplicationStatus），如以下示例响应所示。

```
{
   "AcceptRanges":"bytes",
   "ContentType":"image/jpeg",
   "LastModified":"Mon, 23 Mar 2015 21:02:29 GMT",
   "ContentLength":3191,
   "ReplicationStatus":"COMPLETED",
   "VersionId":"jfnW.HIMOfYiD_9rGbSkmroXsFj3fqZ.",
   "ETag":"\"6805f2cfc46c0f04559748bb039d69ae\"",
   "Metadata":{

   }
}
```

使用 AWS SDK

以下代码片段分别使用 AWS SDK for Java 和 AWS SDK for .NET 获取复制状态。

Java

```
GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest(bucketName,
 key);
ObjectMetadata metadata = s3Client.getObjectMetadata(metadataRequest);

System.out.println("Replication Status : " +
 metadata.getRawMetadataValue(Headers.OBJECT_REPLICATION_STATUS));
```

.NET

```
GetObjectMetadataRequest getmetadataRequest = new GetObjectMetadataRequest
    {
         BucketName = sourceBucket,
         Key        = objectKey
    };

GetObjectMetadataResponse getmetadataResponse =
 client.GetObjectMetadata(getmetadataRequest);
Console.WriteLine("Object replication status: {0}",
 getmetadataResponse.ReplicationStatus);
```

使用 S3 分批复制以复制现有对象

使用 S3 批量复制，您可以复制以下类型的对象：

- 在复制配置设立之前已经存在的对象
- 以前已复制的对象
- 复制失败的对象

您可以使用批量操作任务来按需复制这些对象。S3 批量复制与实时复制不同，后者连续自动跨 Amazon S3 存储桶复制新对象。

要开始使用批量复制，您可以：

- 为新的复制规则或目标启动批量复制 – 您可以在新复制配置中创建第一条规则时，或通过 Amazon S3 控制台向现有配置添加新目标时，创建一次性批量复制任务。
- 为现有复制配置启动批量复制 – 您可以通过 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API，使用 S3 批量操作创建新的批量复制任务。

分批复制任务完成后，您将收到完成报告。有关如何使用报告检查任务的更多信息，请参阅 跟踪任务状态和完成报告。

S3 分批复制注意事项

- 您的源存储桶必须具有现有的复制配置。要启用复制，请参阅 设置实时复制 和 配置实时复制的示例。
- 如果您为存储桶配置了 S3 生命周期，我们建议在批量复制任务处于活动状态时禁用生命周期规则。这样做有助于确保源存储桶和目标存储桶的对等。否则，这些存储桶可能会出现差异，目标存储桶将不会与源存储桶完全一样。例如，考虑以下情形：
  - 您的源存储桶有某个对象的多个版本，并且对于该对象有一个删除标记。
  - 您的源存储桶和目标存储桶具有移除已过期删除标记的生命周期配置。

  在这种情况下，批量复制可能会在复制对象版本之前，将删除标记复制到目标存储桶。这种行为可能导致在复制对象版本之前，生命周期配置将删除标记标为过期，并且从目标存储桶中移除删除标记。
- 您指定用于运行批量操作任务的 AWS Identity and Access Management（IAM）角色必须具有必要的权限，来执行基础的批量复制操作。有关创建 IAM 角色的更多信息，请参阅 配置 IAM 策略以进行分批复制。

- 批量复制需要可以由 Amazon S3 生成的清单。生成的清单必须存储在与源存储桶相同的 AWS 区域中。如果您选择不生成清单，则可以提供包含要复制的对象的 Amazon S3 清单报告或 CSV 文件。
- 批量复制不支持重新复制从目标存储桶中使用对象的版本 ID 删除的对象。要重新复制这些对象，您可以使用分批复制任务将源对象复制到位。将这些对象复制到位会在源存储桶中创建对象的新版本，并自动启动到目标存储桶的复制。删除和重新创建目标存储桶不会启动复制。

  有关批量复制的更多信息，请参阅使用分批操作复制对象的示例。
- 如果您在 S3 存储桶上使用复制规则，请确保通过向附加到复制规则的 IAM 角色授予复制对象的适当权限，来更新您的复制配置。此 IAM 角色必须具有在源存储桶和目标存储桶上执行复制所需的权限。
- 如果您在短时间范围内为同一个存储桶提交多个批量复制任务，Amazon S3 将同时运行这些任务。
- 如果您为两个不同的存储桶提交多个批量复制任务，请注意 Amazon S3 可能无法同时运行所有任务。如果您超过了账户中一次可以运行的批量复制任务的数量，Amazon S3 将暂停优先级较低的任务，来处理优先级较高的任务。优先级较高的项目完成后，任何暂停的任务都将再次变为活动状态。
- 存储在 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类中的对象不支持批量复制。
- 要批量复制存储在归档访问存储层或深度归档访问存储层中的 S3 Intelligent-Tiering 对象，必须首先启动还原请求，然后等待对象移动到频繁访问层。

为分批复制任务指定清单

清单是一种 Amazon S3 对象，其中包含您希望 Amazon S3 采取操作的对象键。如果您希望创建批量复制任务，则必须提供用户生成的清单，或者让 Amazon S3 根据复制配置生成清单。

如果您提供用户生成的清单，清单必须采用 Amazon S3 清单报告或 CSV 文件的形式。如果清单中的对象位于受版本控制的存储桶中，则必须指定对象的版本 ID。只会复制在清单中指定了版本 ID 的对象。要了解有关指定清单的更多信息，请参阅 指定清单。

如果您选择让 Amazon S3 代表您生成清单文件，那么列出的对象将使用与源存储桶的所有复制配置相同的源存储桶、前缀和标签。使用生成的清单，Amazon S3 将复制所有符合条件的对象版本。

> **Note**
>
> 如果您选择让 Amazon S3 生成清单，则清单必须存储在与源存储桶相同的 AWS 区域中。

分批复制任务的筛选条件

在创建批量复制任务时，您可以选择指定其它筛选条件，例如对象创建日期和复制状态，来缩小任务的范围。

您可以通过提供以下一个或多个值基于 ObjectReplicationStatuses 值筛选要复制的对象：

- "NONE" – 表示 Amazon S3 之前从未尝试过复制对象。
- "FAILED" – 表示 Amazon S3 之前尝试过，但未成功复制对象。
- "COMPLETED" – 表示 Amazon S3 之前已成功复制对象。
- "REPLICA" – 表示这是 Amazon S3 已从另一个源复制的副本对象。

有关复制状态的更多信息，请参阅 获取复制状态信息。

如果您不筛选批量复制任务，批量操作将尝试复制清单中与复制配置中的规则匹配的所有对象（无论其 ObjectReplicationStatus 如何），但默认情况下未复制的某些对象除外。有关更多信息，请参阅 the section called “使用复制配置不会复制什么？”

根据您的目标，您可以将 ObjectReplicationStatuses 设置为以下一个或多个值：

- 要复制从未复制过的现有对象，请仅包括 "NONE"。
- 要重试复制之前未能复制的对象，请仅包括 "FAILED"。
- 要同时复制现有对象和重试复制之前未能复制的对象，请同时包括 "NONE" 和 "FAILED"。
- 要使用已复制到另一个目标的对象回填目标存储桶，请包括 "COMPLETED"。
- 要复制之前已复制的对象，请包括 "REPLICA"。

分批复制完成报告

在创建批量复制任务时，您可以请求 CSV 完成报告。此报告显示对象、复制成功或失败代码、输出和描述。有关任务跟踪和完成报告的更多信息，请参阅完成报告。

有关复制故障代码和描述的列表，请参阅 Amazon S3 复制失败原因。

有关排查批量复制故障的信息，请参阅批量复制错误。

批量复制入门

要了解有关如何使用批量复制的更多信息，请参阅教程：使用 S3 批量复制来复制 Amazon S3 存储桶中的现有对象。

配置 IAM 策略以进行分批复制

由于 S3 分批复制是批量操作任务的一种类型，因此您必须创建批量操作 AWS Identity and Access Management（IAM）角色，用于向 Amazon S3 授予代表您执行操作的权限。您还必须将分批复制 IAM 策略附加到批量操作 IAM 角色。以下示例创建一个 IAM 角色，该角色授予批量操作启动分批复制任务的权限。

创建 IAM 角色和策略

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 Access management（访问管理）下，请选择 Roles（角色）。
3. 请选择 Create Role(创建角色)。
4. 请选择 AWS 服务 作为受信任实体的类型，Amazon S3 作为服务，以及 S3 Batch Operations（S3 批量操作）作为使用案例。
5. 请选择下一步: 权限。
6. 请选择创建策略。
7. 请选择 JSON，然后根据清单插入以下策略之一。

> **Note**
>
> 如果您要生成清单或提供清单，则需要不同的权限。有关更多信息，请参阅为分批复制任务指定清单。

使用和存储 S3 生成的清单时的策略

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
```

```
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*",
            "arn:aws:s3:::*** manifest bucket ****/*"
         ]
      }
   ]
}
```

使用用户提供的清单时的策略

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
```

```
            "Effect":"Allow",
            "Resource":[
               "arn:aws:s3:::*** replication source bucket ***/*"
            ]
         },
               {
            "Action":[
               "s3:GetObject",
               "s3:GetObjectVersion"
            ],
            "Effect":"Allow",
            "Resource":[
               "arn:aws:s3:::*** manifest bucket ***/*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:PutObject"
            ],
            "Resource":[
               "arn:aws:s3:::*** completion report bucket ****/*"
            ]
         }
      ]
}
```

8. 请选择下一步：标签。
9. 请选择下一步：审核。
10. 为策略选择名称，然后选择 Create policy（创建策略）。
11. 将此策略附加到你的角色，然后选择 Next: Tags（下一步：标签）。
12. 请选择下一步：审核。
13. 请选择角色的名称，然后选择 Create role（创建角色）。

验证信任策略

1. 登录 AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 请在 Access management（访问管理）下，选择 Roles（角色），然后选择您新创建的角色。

3. 在 Trust relationships（信任关系）选项卡中，请选择 Edit trust relationship（编辑信任关系）。
4. 验证此角色是否使用以下信任策略：

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

为第一个复制规则或新目标创建分批复制任务

在新的复制配置中创建第一个规则或通过 AWS Management Console将新目标添加到现有配置时，您可以选择创建分批复制任务。

要将分批复制用于现有配置而不添加新目标，请参阅为现有复制规则创建分批复制任务。

通过 AWS Management Console 将分批复制用于新复制规则或目标

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择包含您想要复制的对象的存储桶的名称。
3. 要创建新的复制规则或编辑现有规则，请选择 Management（管理），然后向下滚动到 Replication rules（复制规则）：
   - 要创建新的复制规则，请选择 Create replication rule（创建复制规则）。

     > **Note**
     >
     > 有关如何设置基本复制规则的示例，请参阅 配置实时复制的示例。

   - 要编辑现有复制规则，请选择该规则，然后选择 Edit rule（编辑规则）。
4. 创建新的复制规则或编辑现有复制规则的目标，然后选择 Save（保存）。

在新的复制配置中创建第一个规则或编辑现有配置以添加新目标后，将显示 Replicate existing objects?（是否复制现有对象？）对话框，让您可以选择创建分批复制任务。

5. 如果要立即运行此任务，请选择是，复制现有对象。

   如果要以后运行此任务，请选择否，不复制现有对象。

6. 创建您的 S3 分批复制任务。S3 分批复制任务有几个设置：

   任务运行选项

   如果您想让 S3 分批复制任务立即运行，可以选择 Job runs automatically when ready（任务准备就绪后自动运行）。如果您想在以后运行作业，请请选择 Job waits to be run when ready（任务准备就绪后等待运行）。

   如果选择 Job runs automatically when ready（任务在准备就绪后自动运行），您将无法创建和保存批量操作清单。要保存批量操作清单，请选择 Job waits to be run when ready（任务在准备就绪后等待运行）。

   批量操作清单

   清单是您希望对其运行指定操作的所有对象的列表。您可以选择保存批量操作清单。与 S3 清单文件类似，清单将另存为 CSV 文件并存储在存储桶中。要了解有关批量操作清单的更多信息，请参阅 指定清单。

   完成报告

   S3 批量操作将为清单中指定的每个对象执行一个任务。完成报告提供了以合并格式查看任务结果的简单途径，且无需进行任何附加设置。您可以为全部任务或仅为失败的任务请求完成报告。要了解有关完成报告的更多信息，请参阅 完成报告。

   权限

   复制失败的最常见原因之一是提供的 AWS Identity and Access Management (IAM) 角色的权限不足。有关创建此角色的信息，请参阅 配置 IAM 策略以进行分批复制。

7. 请选择 Create Batch Operations job（创建批量操作任务）。

为现有复制规则创建分批复制任务

您可以通过使用 AWS SDK、AWS Command Line Interface（AWS CLI）或 Amazon S3 控制台为现有复制配置来配置 S3 分批复制。有关分批复制的概述，请参阅 使用 S3 分批复制以复制现有对象。

作为先决条件，您必须创建批量操作 AWS Identity and Access Management（IAM）角色，以向 Amazon S3 授予代表您执行操作的权限，请参阅 配置 IAM 策略以进行分批复制。

分批复制任务完成后，您将收到完成报告。有关如何使用报告检查任务的更多信息，请参阅 跟踪任务状态和完成报告。

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Amazon S3 控制台的导航窗格中选择 Batch Operations（批量操作）。
3. 请选择 Create job (创建任务)。
4. 请选择要在其中创建任务的区域。
5. 请选择 Manifest format（清单格式）。此示例将展示如何基于现有 S3 复制配置创建清单。

   > **Note**
   >
   > 清单是您希望对其运行指定操作的所有对象的列表。要了解有关批量操作清单的更多信息，请参阅 指定清单。如果您准备了清单，请选择 S3 清单报告 (manifest.json) 或 CSV。如果清单中的对象位于受版本控制的存储桶中，则应指定对象的版本 ID。有关创建清单的更多信息，请参阅 指定清单。

6. 要根据复制配置创建清单，请选择 Create manifest using S3 Replication configuration（使用 S3 复制配置创建清单）。然后，选择您的复制配置的源存储桶。
7. （可选）您可以包括其他筛选条件，例如对象创建日期和复制状态。有关如何按复制状态筛选的示例，请参阅 为分批复制任务指定清单。
8. 要保存清单，请选择 Save Batch Operations manifest（保存批量操作清单）。

   a. 如果您选择生成和保存清单，则必须选择此账户中的存储桶或另一个 AWS 账户 中的存储桶。请在文本框中指定存储桶名称。

      > **Note**
      >
      > 生成的清单必须存储在与源存储桶相同的 AWS 区域 中。

   b. 请选择加密类型。
9. （可选）提供 Description（说明）。

10. 必要时调整任务的 Priority（优先级）。数字越大，优先级越高。Amazon S3 尝试在优先级较低的任务之前运行优先级较高的任务。有关任务优先级的更多信息，请参阅 分配任务优先级。
11. （可选）生成完成报告。要生成报告，请选择 Generate completion report（生成完成报告）。

    如果您选择生成完成报告，则必须选择报告仅失败的任务或全部任务，并为报告提供目标存储桶。
12. 选择有效的 IAM 角色。

    > **Note**
    >
    > 有关创建 IAM 角色的更多信息，请参阅 配置 IAM 策略以进行分批复制。

13. （可选）将任务标签添加到分批复制任务中。
14. 请选择 Next (下一步)。
15. 检查您的任务配置，然后选择 Create job（创建任务）。

通过 S3 清单使用 AWS CLI

以下示例使用 S3 为 AWS 账户 *111122223333* 生成的清单创建 S3 分批复制任务。此示例将尝试复制现有对象和之前未能复制的对象。有关按复制状态筛选的信息，请参阅 为分批复制任务指定清单。

```
aws s3control create-job --account-id 111122223333 --operation
 '{"S3ReplicateObject":{}}' --report '{"Bucket":"arn:aws:s3:::***
 completion report bucket ****","Prefix":"batch-replication-report",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}'
 --manifest-generator '{"S3JobManifestGenerator": {"ExpectedBucketOwner":
 "111122223333", "SourceBucket": "arn:aws:s3:::*** replication source bucket
 ***", "EnableManifestOutput": false, "Filter": {"EligibleForReplication": true,
 "ObjectReplicationStatuses": ["NONE","FAILED"]}}}' --priority 1 --role-arn
 arn:aws:iam::111122223333:role/batch-Replication-IAM-policy --no-confirmation-required
 --region source-bucket-region
```

> **Note**
>
> 该任务必须从同一个 AWS 区域 复制源存储桶中启动。IAM 角色 `role/batch-Replication-IAM-policy` 之前已创建。请参阅 配置 IAM 策略以进行分批复制。

成功启动分批复制任务后，您会收到任务 ID 作为响应。您可以使用下面的命令监控此任务。

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-
bucket-region
```

通过用户提供的清单使用 AWS CLI

以下示例使用 AWS 账户 *111122223333* 的用户定义清单创建 S3 分批复制任务。如果清单中的对象位于受版本控制的存储桶中，则必须指定对象的版本 ID。只有在清单中指定了版本 ID 的对象才会复制。有关创建清单的更多信息，请参阅 指定清单。

```
aws s3control create-job --account-id 111122223333 --operation
 '{"S3ReplicateObject":{}}' --report '{"Bucket":"arn:aws:s3:::***
 completion report bucket ****","Prefix":"batch-replication-report",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}'
 --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key","VersionId"]},"Location":{"ObjectArn":"arn:aws:s3:::*** completion
 report bucket ****/manifest.csv","ETag":"Manifest Etag"}}' --priority 1 --role-arn
 arn:aws:iam::111122223333:role/batch-Replication-IAM-policy --no-confirmation-required
 --region source-bucket-region
```

> **Note**
>
> 该任务必须从同一个 AWS 区域 复制源存储桶中启动。IAM 角色 `role/batch-Replication-IAM-policy` 之前已创建。请参阅 配置 IAM 策略以进行分批复制。

成功启动分批复制任务后，您会收到任务 ID 作为响应。您可以使用下面的命令监控此任务。

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-
bucket-region
```

使用标签对存储进行分类

使用对象标签对存储进行分类。每个标签都是一个键-值对。

您可以将标签添加到新对象（当您上传新对象时），也可以将标签添加到现有对象。

- 您最多可以将 10 个标签与对象关联。与对象关联的标签必须具有唯一的标签键。

- 标签键的长度最大可以为 128 个 Unicode 字符，标签值的长度最大可以为 256 个 Unicode 字符。Amazon S3 对象标签在内部以 UTF-16 表示。请注意，采用 UTF-16 时，字符占用 1 或 2 个字符位置。
- 键和值区分大小写。
- 有关标签限制的更多信息，请参阅《AWS 账单与成本管理用户指南》中的 User-defined tag restrictions。有关基本标签限制，请参阅《Amazon EC2 用户指南》中的标签限制。

示例

请考虑以下标签示例：

Example PHI 信息

假设某个对象包含受保护医疗信息 (PHI) 数据。您可以使用以下键/值对标记该对象。

```
PHI=True
```

或

```
Classification=PHI
```

Example 项目文件

假设您将项目文件存储在 S3 桶中。您可以使用一个名为 `Project` 的键和一个值标记这些对象，如下所示。

```
Project=Blue
```

Example 多个标签

您可以将多个标签添加到一个对象，如下所示。

```
Project=x
Classification=confidential
```

键名前缀和标签

您还可以使用对象键名前缀来分类存储。但是，基于前缀的分类是一维的。请考虑以下对象键名：

```
photos/photo1.jpg
project/projectx/document.pdf
project/projecty/document2.pdf
```

这些键名具有前缀 photos/、project/projectx/ 和 project/projecty/。这些前缀支持一维分类。即，一个前缀下的一切都属于一个类别。例如，前缀 project/projectx 可确定与项目 x 相关的所有文档。

利用标签，您现在获得了另一个维度。如果您希望 photo1 属于项目 x 类别，则可以相应地标记该对象。

其他优势

除了数据分类之外，标签还提供了下列好处：

- 对象标签支持权限的精细访问控制。例如，您可以向用户授予仅读取带有特定标签的对象的权限。
- 对象标签支持精细的对象生命周期管理，在其中，除了在生命周期规则中指定键名前缀之外，还可以指定基于标签的筛选条件。
- 使用 Amazon S3 分析时，您可以配置筛选条件，以便按对象标签、键名前缀或前缀和标签的组合对对象进行分组以进行分析。
- 您还可以自定义 Amazon CloudWatch 指标以按特定标签筛选条件显示信息。以下各节提供了详细信息。

> **Important**
>
> 使用标签来标记包含机密数据（如 personally identifiable information (PII) 或受保护医疗信息 (PHI)）的对象是可以接受的。但标签本身不应包含任何机密信息。

通过单个请求将对象标签集添加到多个 Amazon S3 对象

要使用单个请求向多个 Amazon S3 对象添加对象标签集，您可以使用 S3 分批操作。您为 S3 批量操作提供要操作的对象列表。S3 批量操作调用相应的 API 操作来执行指定的操作。单个批量操作任务可对包含 EB 级数据的数十亿个对象执行指定操作。

S3 批量操作特征包括跟踪进度、发送通知并存储所有操作的详细完成报告，从而提供完全托管、可审核的无服务器体验。您可以通过 Amazon S3 控制台、AWS CLI、AWS SDK 或 REST API 使用 S3 批量操作。有关更多信息，请参阅 the section called “批量操作基础知识”。

有关对象标签的更多信息，请参阅 管理对象标签。

与对象标签相关的 API 操作

Amazon S3 支持特定于对象标签的以下 API 操作：

对象 API 操作

- PUT 对象标签 – 替换对象上的标签。您可以在请求正文中指定标签。使用此 API 的对象标签管理有两个不同的情形。
  - 对象没有标签 - 利用此 API，您可以将一组标签添加到某个对象 (该对象没有以前的标签)。
  - 对象有一组现有标签 - 要修改现有标签，您必须先检索现有标签集，在客户端侧修改它，然后使用此 API 替换它。

    > **Note**
    >
    > 如果您发送带有空标签集的此请求，Amazon S3 将删除对象上的现有标签集。如果您使用此方法，则将需为套餐 1 请求 (PUT) 付费。有关更多信息，请参阅 Amazon S3 定价。DELETE 对象标签请求优先进行，因为它可获得相同的结果而不会产生费用。

- GET 对象标签 – 返回与某个对象关联的标签集。Amazon S3 将在响应正文中返回对象标签。
- DELETE 对象标签 – 删除与某个对象关联的标签集。

支持标签的其他 API 操作

- PUT 对象和启动分段上传 – 您可以在创建对象时指定标签。您将使用 `x-amz-tagging` 请求标头指定标签。
- GET 对象 – Amazon S3 不会返回标签集，而会在 `x-amz-tag-count` 标头中返回对象标签计数（仅当请求者有权读取标签时），因为标头响应大小限制为 8000 个字节。如果要查看标签，您应该再发出一个 GET 对象标签 API 操作请求。
- POST 对象 – 您可以在 POST 请求中指定标签。

  只要请求中的标签不超过 8 K 个字节的 HTTP 请求标头大小限制，您就可以使用 PUT Object API 创建带标签的对象。如果您指定的标签超过了标头大小限制，您可以使用将标签包含在正文中的此 POST 方法。

  PUT 对象 – 复制 – 您可以在请求中指定 `x-amz-tagging-directive` 以指示 Amazon S3 复制（默认行为）标签或将标签替换为请求中提供的一组新标签。

请注意以下几点：

- S3 对象标签非常一致。有关更多信息，请参阅 Amazon S3 数据一致性模型。

其他配置

本节介绍对象标签如何与其他配置关联。

对象标签和生命周期管理

在桶生命周期配置中，您可以指定筛选条件以选择该规则适用的一部分对象。您可以基于键名前缀、对象标签或两者指定筛选条件。

假设您将照片（原始格式和已完成格式）存储在 Amazon S3 桶中。您可以按下面所示标记这些对象。

```
phototype=raw
or
phototype=finished
```

您可能考虑有时候在创建原始照片后将其存档到 S3 Glacier。您可以在生命周期规则中配置一个筛选条件，用于确定一部分包含特定标签 (photos/) 且带有键名称前缀 (phototype=raw) 的对象。

有关更多信息，请参阅 管理存储生命周期。

对象标签和复制

如果您在桶上配置了复制，Amazon S3 将复制标签，前提是您向 Amazon S3 授予了读取标签的权限。有关更多信息，请参阅 设置实时复制。

对象标记事件通知

您可以设置 Amazon S3 事件通知以在添加对象标签或从对象中删除时接收通知。`s3:ObjectTagging:Put` 事件类型会在对象上放置标签或现有标签更新时通知您。当从对象中移除标签时，`s3:ObjectTagging:Delete` 事件类型会通知您。有关更多信息，请参阅 Enabling event notifications（启用事件通知）。

有关对象标签的更多信息，请参阅以下主题：

主题

- 标签和访问控制策略
- 管理对象标签

标签和访问控制策略

您还可以使用权限策略（桶和用户策略）管理对象标签相关权限。有关策略操作，请参阅以下主题：

- 对象操作
- 存储桶操作

对象标签支持用于管理权限的精细访问控制。您可以基于对象标签授予条件权限。Amazon S3 支持以下条件键，这些键可用于授予基于对象标签的条件权限：

- `s3:ExistingObjectTag/<tag-key>` - 使用此条件键可验证现有对象标签是否有特定标签键和值。

  > **Note**
  >
  > 当授予 PUT Object 和 DELETE Object 操作的权限时，此条件键不受支持。也就是说，您无法创建这样一个策略：允许或拒绝用户基于对象的现有标签删除或覆盖该对象。

- `s3:RequestObjectTagKeys` - 使用此条件键可限制要在对象上允许的标签键。当使用 PutObjectTagging 和 PutObject 以及 POST 对象请求将标签添加到对象时，这很有用。
- `s3:RequestObjectTag/<tag-key>` - 使用此条件键可限制要在对象上允许的标签键和值。当使用 PutObjectTagging 和 PutObject 以及 POST 桶请求将标签添加到对象时，这很有用。

有关特定于 Amazon S3 服务的条件键的完整列表，请参阅 使用条件键的存储桶策略示例。以下权限策略说明了对象标签如何支持精细访问权限管理。

Example 1：允许用户仅读取具有特定标签和键值的对象

以下权限策略限制用户只能读取具有 `environment: production` 标签键和值的对象。请注意，该策略使用 `s3:ExistingObjectTag` 条件键来指定标签键和值。

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Principal": {
      "AWS": [
        "arn:aws:iam::111122223333:role/JohnDoe"
```

```
      ]
    },
    "Effect": "Allow",
    "Action": ["s3:GetObject", "s3:GetObjectVersion"],
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
    "Condition": {
      "StringEquals":
        {"s3:ExistingObjectTag/environment": "production"}
    }
  }
  ]
}
```

Example 2：限制用户可以添加哪些对象标签键

以下权限策略将向用户授予执行 s3:PutObjectTagging 操作的权限，这使用户可以将标签添加到现有对象。条件使用 s3:RequestObjectTagKeys 条件键指定允许的标签键，例如 Owner 或 CreationDate。有关更多信息，请参阅《IAM 用户指南》中的创建测试多个键值的条件。

该策略确保在请求中指定的每个标签键都是授权的标签键。条件中的 ForAnyValue 限定符确保请求中必须至少存在指定的值之一。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
```

```
  ]
}
```

Example 3：在允许用户添加对象标签时需要特定的标签键和值

以下示例策略将向用户授予执行 s3:PutObjectTagging 操作的权限，这使用户可以将标签添加到现有对象。条件要求用户包含值设置为 *X* 的特定标签键（如 *Project*）。

```
{
   "Version": "2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
         ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

管理对象标签

本节介绍如何使用适用于 Java 和 .NET 的 AWS 开发工具包或 Amazon S3 控制台管理对象标签。

对象标签为您提供了对存储进行分类的方法。每个标签都是遵循以下规则的键值对：

- 您最多可以将 10 个标签与对象关联。与对象关联的标签必须具有唯一的标签键。
- 标签键的长度最大可以为 128 个 Unicode 字符，标签值的长度最大可以为 256 个 Unicode 字符。Amazon S3 对象标签在内部以 UTF-16 表示。请注意，采用 UTF-16 时，字符占用 1 或 2 个字符位置。
- 键和值区分大小写。

有关对象标签的更多信息，请参阅 使用标签对存储进行分类。有关标签限制的更多信息，请参阅《AWS Billing and Cost Management 用户指南》中的用户定义的标签限制。

使用 S3 控制台

向对象添加标签

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在桶列表中，选择包含要向其添加标签的对象的桶的名称。

   您还可以选择导航到文件夹。
3. 在对象列表中，选中要向其添加标签的对象名称旁边的复选框。
4. 在操作菜单上选择编辑标签。
5. 查看列出的对象，然后选择添加标签。
6. 每个对象标签都是一个键值对。输入 Key (键) 和 Value (值)。要添加另一个标签，请选择 Add Tag (添加标签)。

   您最多可以为一个对象输入 10 个标签。
7. 选择保存更改。

   Amazon S3 会将标签添加给指定对象。

有关更多信息，另请参阅本指南中的 在 Amazon S3 控制台中查看对象属性 和 上传对象。

使用 AWS 开发工具包

Java

以下示例演示如何使用AWS SDK for Java为新对象设置标签并检索或替换现有对象的标签。有关对象标签的更多信息，请参阅使用标签对存储进行分类。有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

```
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class ManagingObjectTags {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Object key ***";
        String filePath = "*** File path ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Create an object, add two new tags, and upload the object to Amazon
 S3.
            PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
 new File(filePath));
            List<Tag> tags = new ArrayList<Tag>();
            tags.add(new Tag("Tag 1", "This is tag 1"));
            tags.add(new Tag("Tag 2", "This is tag 2"));
            putRequest.setTagging(new ObjectTagging(tags));
            PutObjectResult putResult = s3Client.putObject(putRequest);

            // Retrieve the object's tags.
            GetObjectTaggingRequest getTaggingRequest = new
 GetObjectTaggingRequest(bucketName, keyName);
            GetObjectTaggingResult getTagsResult =
 s3Client.getObjectTagging(getTaggingRequest);

            // Replace the object's tags with two new tags.
            List<Tag> newTags = new ArrayList<Tag>();
            newTags.add(new Tag("Tag 3", "This is tag 3"));
            newTags.add(new Tag("Tag 4", "This is tag 4"));
            s3Client.setObjectTagging(new SetObjectTaggingRequest(bucketName,
 keyName, new ObjectTagging(newTags)));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
```

```
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

以下示例演示如何使用AWS SDK for .NET为新对象设置标签并检索或替换现有对象的标签。有关对象标签的更多信息，请参阅使用标签对存储进行分类。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    public class ObjectTagsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for the new object ***";
        private const string filePath = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            PutObjectWithTagsTestAsync().Wait();
        }
```

```
        static async Task PutObjectWithTagsTestAsync()
        {
            try
            {
                // 1. Put an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>{
                        new Tag { Key = "Keyx1", Value = "Value1"},
                        new Tag { Key = "Keyx2", Value = "Value2" }
                    }
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);
                // 2. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest = new GetObjectTaggingRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);
                for (int i = 0; i < objectTags.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
objectTags.Tagging[i].Key, objectTags.Tagging[i].Value);


                // 3. Replace the tagset.

                Tagging newTagSet = new Tagging();
                newTagSet.TagSet = new List<Tag>{
                    new Tag { Key = "Key3", Value = "Value3"},
                    new Tag { Key = "Key4", Value = "Value4" }
                };


                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
```

```
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet
                };
                PutObjectTaggingResponse response2 = await
 client.PutObjectTaggingAsync(putObjTagsRequest);

                // 4. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest2 = new
 GetObjectTaggingRequest();
                getTagsRequest2.BucketName = bucketName;
                getTagsRequest2.Key = keyName;
                GetObjectTaggingResponse objectTags2 = await
 client.GetObjectTaggingAsync(getTagsRequest2);
                for (int i = 0; i < objectTags2.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
 objectTags2.Tagging[i].Key, objectTags2.Tagging[i].Value);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an
 object"
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Encountered an error. Message:'{0}' when writing an object"
                    , e.Message);
            }
        }
    }
}
```

使用成本分配 S3 存储桶标签

要跟踪单个项目或项目组的存储成本或其他标准，请使用成本分配标签标记您的 Amazon S3 存储桶。成本分配标签是与 S3 存储桶相关联的键-值对。在激活成本分配标签后，AWS 将使用这些标签在

成本分配报告上整理您的资源成本。成本分配标签只能用于标记存储桶。有关用于标记对象的标签的信息，请参阅使用标签对存储进行分类。

成本分配报告会按产品分类和关联的账户用户列出您账户的 AWS 使用量。该报告包含与详细账单报告相同的行项目 (请参阅了解 Amazon S3 的 AWS 账单和使用情况报告) 和用于您的标签键的附加列。

AWS 提供了两种类型的成本分配标签，即 AWS 生成的标签和用户定义的标签。AWS 在 Amazon S3 CreateBucket 事件之后为您定义、创建和应用 AWS 生成的 `createdBy` 标签。您定义、创建用户定义的 标签并将其应用到 S3 存储桶。

必须在账单和成本管理控制台中分别激活这两种类型的标签，然后它们才能出现在账单报告中。有关 AWS 生成的标签的更多信息，请参阅 AWS 生成的成本分配标签。

- 要在控制台中创建标签，请参阅查看 S3 存储桶的属性。
- 要使用 Amazon S3 API 创建标签，请参阅《Amazon Simple Storage Service API 参考》中的 PUT 存储桶标记。
- 要使用 AWS CLI 创建标签，请参阅《AWS CLI 命令参考》中的 put-bucket-tagging。
- 有关激活标签的更多信息，请参阅《AWS Billing 用户指南》中的使用成本分配标签。

用户定义的成本分配标签

用户定义的成本分配标签有以下组成部分：

- 标签键。标签键是标签的名称。例如，在标签 project/Trinity 中，project 是键。标记键是一个区分大小写的字符串，它可以包含 1 到 128 个 Unicode 字符。

- 标签值。标签值是必需的字符串。例如，在标签 project/Trinity 中，Trinity 是值。标记值是一个区分大小写的字符串，它可以包含 0 到 256 个 Unicode 字符。

有关用户定义的标签的允许字符以及其他限制的详细信息，请参阅《AWS Billing 用户指南》中的用户定义的标签限制。有关用户定义的标签的更多信息，请参阅《AWS Billing 用户指南》中的用户定义的成本分配标签。

S3 存储桶标签

每个 S3 存储桶都有一个标签集。标签集包含了分配到该存储桶的所有标签。一个标签集可以包含多达 50 个标签，也可以为空。键在标签集中必须是唯一的，但标签集中的值不必是唯一的。例如，您可以在名为 project/Trinity 和 cost-center/Trinity 的标签集中使用相同的值。

在存储桶内，如果您要添加的标签与现有标签的键相同，则新值会覆盖旧值。

AWS不会对您的标签应用任何语义意义。我们严格按字符串解释标签。

要添加、列出、编辑或删除标签，您可以使用 Amazon S3 控制台、AWS Command Line Interface (AWS CLI) 或 Amazon S3 API。

更多信息

- 《AWS Billing 用户指南》中的使用成本分配标签
- 了解 Amazon S3 的 AWS 账单和使用情况报告
- Amazon S3 的 AWS Billing报告

Amazon S3 的账单和使用情况报告

当使用 Amazon S3 时，您不必支付任何前期费用或承诺您将存储多少内容。与其它 AWS 服务一样，您按实际使用量付费，只需为所用的资源付费。

AWS 提供 Amazon S3 的以下报告：

- 账单报告 – 包含多份报告，旨在提供您正在使用的 AWS 服务（包括 Amazon S3）的所有活动的详细信息。AWS 始终向 S3 存储桶拥有者收取 Amazon S3 费用，除非该桶是作为申请方付款桶创建的。有关请求者付费的更多信息，请参阅 使用申请方付款存储桶进行存储传输和使用。有关账单报告的更多信息，请参阅 Amazon S3 的 AWS Billing报告。
- 使用情况报告 – 概括了特定服务按小时、天或月聚合的活动情况。可以选择要包括的使用类型和操作。您还可以选择数据聚合的方式。有关更多信息，请参阅 AWSAmazon S3 的使用情况报告。

以下主题提供有关 Amazon S3 账单和使用情况报告的信息。

主题

- Amazon S3 的 AWS Billing报告
- AWSAmazon S3 的使用情况报告
- 了解 Amazon S3 的 AWS 账单和使用情况报告
- Amazon S3 错误响应的账单计费

Amazon S3 的 AWS Billing报告

AWS 为您提供的月度账单按 AWS 服务和功能分隔您的使用情况信息和成本。有多种 AWS Billing报告可用：月度报告、成本分配报告和详细账单报告。有关如何查看您的账单报告的信息，请参阅《AWS Billing 用户指南》中的查看您的账单。

要跟踪您的 AWS 使用情况并提供与您的账户关联的估算费用，您可以设置 AWS Cost and Usage Reports。有关更多信息，请参阅《AWS Data Exports Guide》中的 What are AWSCost and Usage Reports?

还可以下载使用情况报告，它会比账单报告给出更多有关 Amazon S3 存储使用量的详细信息。有关更多信息，请参阅 AWSAmazon S3 的使用情况报告。

下表列出了与 Amazon S3 使用率相关联的费用。

Amazon S3 使用费

费用	注释
存储	您要为在您的 S3 存储桶中存储对象付费。您付费的费率取决于对象的大小、当月存储对象的时长以及存储类。Amazon S3 提供以下存储类：S3 Standard、S3 Express One Zone、S3 Intelligent-Tiering、S3 Standard-IA（IA 表示不频繁访问）、S3 One Zone-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive 或低冗余存储（RRS）。有关存储类的更多信息，请参阅 使用 Amazon S3 存储类。 请注意，如果启用了 S3 版本控制，则您需要为保留的对象的每个版本付费。有关版本控制的更多信息，请参阅S3 版本控制的工作原理。
监控和自动化	您需要按月为 S3 Intelligent-Tiering 存储类中存储的每个对象支付监控和自动化费用，以便监控访问模式和在 S3 Intelligent-Tiering 中的访问层之间移动对象。

费用	注释
请求	您要为请求付费，例如，针对您的 S3 存储桶和对象进行的 GET 请求。这包括生命周期请求。请求的费率取决于您发出的请求类型。有关请求定价的信息，请参阅 Amazon S3 定价。
检索	您要为检索存储在 S3 Standard-IA、S3 One Zone-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储中的对象付费。
提前删除	如果在最小存储承诺过去之前删除存储在、S3 Standard-IA、S3 One Zone-IA、S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储中的对象，您需要为该对象支付提前删除费用。
存储管理	您需要为在您账户的桶上启用的存储管理特征（Amazon S3 清单、分析和对象标记）支付费用。
带宽	您需要为所有进出 Amazon S3 的带宽支付费用，以下各项除外： • 从 Internet 传入的数据 • 传出到 Amazon Elastic Compute Cloud（Amazon EC2）实例的数据，当实例与 S3 存储桶位于同一 AWS 区域 时 • 传出到 Amazon CloudFront (CloudFront) 的数据 您还需要为使用 Amazon S3 Transfer Acceleration 传输的任何数据支付费用。

有关存储、数据传输和服务的 Amazon S3 使用收费的详细信息，请参阅 Amazon S3 定价和 Amazon S3 常见问题。

有关了解在 Amazon S3 的账单和使用情况报告中使用的代码和缩写的信息，请参阅了解 Amazon S3 的 AWS 账单和使用情况报告。

更多信息

- AWSAmazon S3 的使用情况报告
- 使用成本分配 S3 存储桶标签
- AWS Billing与成本管理
- Amazon S3 定价

AWSAmazon S3 的使用情况报告

在下载使用率报告时，您可以选择按小时、天或月聚合使用量数据。Amazon S3 使用情况报告将按使用类型和 AWS 区域 列示操作。有关您的 Amazon S3 存储使用量的更多详细报告，请下载动态生成的 AWS 使用情况报告。可以选择要包括的使用类型、操作和时间段。您还可以选择数据聚合的方式。有关使用情况报告的更多信息，请参阅《AWS Data Exports User Guide》中的 AWS Usage Report。

Amazon S3 使用情况报告包含以下信息：

- Service – Amazon S3
- Operation – 对您的存储桶或对象执行的操作。有关 Amazon S3 操作的详细说明，请参阅 跟踪使用率报告中的操作。
- UsageType – 以下值之一：
  - 标识存储类型的代码
  - 标识请求类型的代码
  - 标识检索类型的代码
  - 标识数据传输类型的代码
  - 标识从 S3 Intelligent-Tiering、S3 Standard-IA、S3 One Zone-Infrequent Access (S3 One Zone-IA)、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储中进行的提前删除的代码
  - `StorageObjectCount` – 在给定存储桶中存储的对象计数

  有关 Amazon S3 使用类型的详细说明，请参阅 了解 Amazon S3 的 AWS 账单和使用情况报告。
- Resource – 与列出的使用率相关联的存储桶的名称。

- StartTime – 以协调世界时 (UTC) 表示的使用开始时间。
- EndTime – 以协调世界时 (UTC) 表示的使用结束时间。
- UsageValue – 以下容量值之一：数据的典型度量单位是千兆字节 (GB)。但是，根据服务和报告的不同，可能会显示万亿字节 (TB)。
  - 指定时间段内的请求数
  - 传输的数据量
  - 给定小时内存储的数据量
  - 与从 S3 Standard-IA、S3 One Zone-IA、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储中进行的还原相关联的数据量

> **Tip**
>
> 有关 Amazon S3 收到的针对您对象的每个请求的详细信息，请打开您的存储桶的服务器访问日志记录。有关更多信息，请参阅 使用服务器访问日志记录来记录请求。

您可以下载 XML 或逗号分隔值 (CSV) 文件格式的使用率报告。下面是在电子表格应用程序中打开的示例 CSV 使用率报告。

Service	Operation	UsageType	Resource	StartTime	EndTime	UsageValue
AmazonS3	HeadBucket	USW2-C3DataTransfer-Out-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	15309
AmazonS3	PutObject	USW2-C3DataTransfer-In-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	19062
AmazonS3	HeadBucket	USW2-Requests-Tier2	admin-created3	6/1/2017 0:00	7/1/2017 0:00	68
AmazonS3	PutObjectForRepl	USW1-Requests-SIA-Tier1	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	178294
AmazonS3	PutObjectForRepl	USW1-USW2-AWS-In-Bytes	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	387929083
AmazonS3	GetObjectForRepl	USW2-Requests-NoCharge	admin-created3	6/1/2017 0:00	7/1/2017 0:00	108
AmazonS3	GetObjectForRepl	USW2-USW1-AWS-Out-Bytes	my-test-bucket-bash	6/1/2017 0:00	7/1/2017 0:00	387910021

有关更多信息，请参阅 了解 Amazon S3 的 AWS 账单和使用情况报告。

下载 AWS 使用率报告

您可以下载 XML 或 CSV 文件格式的使用情况报告。

下载使用率报告

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在标题栏中，选择您的用户名或账户 ID，然后选择账单与成本管理。

3. 在导航窗格中，选择成本和使用情况报告。
4. 在“AWS 使用情况报告”下，选择创建使用情况报告。
5. 在下载使用情况报告页面上，选择以下设置:
   - 服务 – 选择 Amazon Simple Storage Service。
   - Usage Types (使用类型) – 有关 Amazon S3 使用类型的详细说明，请参阅 了解 Amazon S3 的 AWS 账单和使用情况报告。
   - Operation (操作) – 有关 Amazon S3 操作的详细说明，请参阅 跟踪使用率报告中的操作。
   - Time Period (时间段) – 您希望报告涵盖的时间段。
   - Report Granularity (报告粒度) – 您是否要让报告包含按小时、天或月计算的小计。
6. 选择下载，选择下载格式（XML 报告或 CSV 报告），然后按照提示打开或保存报告。

更多信息

- 了解 Amazon S3 的 AWS 账单和使用情况报告
- Amazon S3 的 AWS Billing报告

了解 Amazon S3 的 AWS 账单和使用情况报告

Amazon S3 账单和使用情况报告使用代码和缩写。对于下表中的使用类型，请使用此列表中的缩写替换 *region*、*region1* 和 *region2*：

- APE1：亚太地区（香港）
- APN1：亚太地区（东京）
- APN2：亚太地区（首尔）
- APN3：亚太地区（大阪）
- APS1：亚太地区（新加坡）
- APS2：亚太地区（悉尼）
- APS3：亚太地区（孟买）
- APS4：亚太地区（雅加达）
- APS5：亚太地区（海得拉巴）
- APS6：亚太地区（墨尔本）
- CAN1：加拿大（中部）

- CAN2：加拿大西部（卡尔加里）
- CNN1：中国（北京）
- CNW1：中国（宁夏）
- AFS1：非洲（开普敦）
- EUC2：欧洲（苏黎世）
- EUN1：欧洲（斯德哥尔摩）
- EUS2：欧洲（西班牙）
- EUC1：欧洲（法兰克福）
- EU：欧洲（爱尔兰）
- EUS1：欧洲（米兰）
- EUW2：欧洲（伦敦）
- EUW3：欧洲（巴黎）
- ILC1：以色列（特拉维夫）
- MEC1：中东（阿联酋）
- MES1：中东（巴林）
- SAE1：南美洲（圣保罗）
- UGW1： AWS GovCloud（美国西部）
- UGE1： AWS GovCloud（美国东部）
- USE1（或无前缀）：美国东部（弗吉尼亚北部）
- USE2：美国东部（俄亥俄）
- USW1：美国西部（加利福尼亚北部）
- USW2：美国西部（俄勒冈）

对于下表中的 S3 多区域接入点使用类型，请使用此列表中的缩写替换 *regiongroup1* 和 *regiongroup2*：

- AP：亚太地区
- AU：澳大利亚
- EU：欧洲
- IN：印度

- NA：北美洲
- SA：南美洲

区域组是由多个 AWS 区域组成的地理分组。有关更多信息，请参阅区域和可用区。有关各 AWS 区域的定价信息，请参阅 Amazon S3 定价。

下表中的第一列列出了您的账单和使用率报告中显示的使用类型。数据的典型度量单位是千兆字节 (GB)。但是，根据服务和报告的不同，可能会显示万亿字节 (TB)。

使用类型

Usage Type	单位	粒度	描述
region1-*region2*-AWS-In-ABytes	GB	每小时	从 *region2* 传输到 *region1* 的加速数据量
region1-*region2*-AWS-In-ABytes-T1	GB	每小时	从 *region2* 传输到 *region1* 的 T1 加速数据量，其中 T1 是向美国、欧洲和日本的节点（POPs）发出的 CloudFront 请求
region1-*region2*-AWS-In-ABytes-T2	GB	每小时	从 *region2* 传输到 *region1* 的 T2 加速数据量，其中 T2 是向所有其它 AWS 边缘站点的 POPs 发出的 CloudFront 请求
region1-*region2*-AWS-In-Bytes	GB	每小时	从 *region2* 传输到 *region1* 的数据量
region1-*region2*-AWS-Out-ABytes	GB	每小时	从 *region1* 传输到 *region2* 的加速数据量
region1-*region2*-AWS-Out-ABytes-T1	GB	每小时	从 *region1* 传输到 *region2* 的 T1 加速数据量，其中 T1 是向美国、

Usage Type	单位	粒度	描述
			欧洲和日本的 POP 发出的 CloudFront 请求
region1-*region2*-AWS-Out-ABytes-T2	GB	每小时	从 *region1* 传输到 *region2* 的 T2 加速数据量，其中 T2 是向所有其它 AWS 边缘站点的 POP 发出的 CloudFront 请求
region1-*region2*-AWS-Out-Bytes	GB	每小时	从 *region1* 传输到 *region2* 的数据量
region-BatchOperations-Jobs	计数	每小时	执行的 S3 批量操作作业数
region-BatchOperations-Objects	计数	每小时	由 S3 批量操作执行的对象操作数
region-Bulk-Retrieval-Bytes	GB	每小时	使用批量 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 请求检索的数据量
region-BytesDeleted-GDA	GB	每月	`DeleteObject` 操作从 S3 Glacier Deep Archive 存储中删除的数据量
region-BytesDeleted-GIR	GB	每月	`DeleteObject` 操作从 S3 Glacier Instant Retrieval 存储中删除的数据量。
region-BytesDeleted-GLACIER	GB	每月	`DeleteObject` 操作从 S3 Glacier Flexible Retrieval 存储中删除的数据量

Usage Type	单位	粒度	描述
region-BytesDeleted-INT	GB	每月	DeleteObject 操作从 S3 Intelligent-Tiering 存储中删除的数据量
region-BytesDeleted-RRS	GB	每月	DeleteObject 操作从低冗余存储（RRS）存储中删除的数据量
region-BytesDeleted-SIA	GB	每月	DeleteObject 操作从 S3 Standard-IA 存储中删除的数据量
region-BytesDeleted-STANDARD	GB	每月	DeleteObject 操作从 S3 Standard 存储中删除的数据量
region-BytesDeleted-ZIA	GB	每月	DeleteObject 操作从 S3 One Zone-IA 存储中删除的数据量
region-C3DataTransfer-In-Bytes	GB	每小时	在同一 AWS 区域 内从 Amazon EC2 传输到 Amazon S3 的数据量
region-C3DataTransfer-Out-Bytes	GB	每小时	在同一 AWS 区域 内从 Amazon S3 传输到 Amazon EC2 的数据量
region-CloudFront-In-Bytes	GB	每小时	从 CloudFront 分配传输到 AWS 区域 的数据量
region-CloudFront-Out-Bytes	GB	每小时	从 AWS 区域 传输到 CloudFront 分配的数据量
region-DataTransfer-In-Bytes	GB	每小时	从 Internet 传输到 Amazon S3 的数据量

Usage Type	单位	粒度	描述
region-DataTransfer-Out-Bytes	GB	每小时	从 Amazon S3 传输到 Internet 的数据量[1]
region-DataTransfer-Regional-Bytes	GB	每小时	在同一 AWS 区域 内从 Amazon S3 传输到 AWS 资源的数据量
region-EarlyDelete-ByteHrs	GB-小时数	每小时	在 90 天最低承诺结束之前，从 S3 Glacier Flexible Retrieval 存储中删除的对象的按比例存储使用量[2]
region-EarlyDelete-GDA	GB-小时数	每小时	在 180 天最低承诺结束之前，从 S3 Glacier Deep Archive 存储中删除的对象的按比例存储使用量2
region-EarlyDelete-GIR	GB-小时数	每小时	在 90 天最低承诺结束之前，从 S3 Glacier Instant Retrieval 中删除的对象的比例存储使用。
region-EarlyDelete-GIR-SmObjects	GB-小时数	每小时	在 90 天最低承诺结束之前，从 S3 Glacier Instant Retrieval 中删除的小对象 (小于 128KB) 的按比例存储使用量3
region-EarlyDelete-SIA	GB-小时数	每小时	在 30 天最低承诺结束之前，从 S3 Standard-IA 中删除的对象的按比例存储使用量[3]

Usage Type	单位	粒度	描述
region-EarlyDelete-SIA-SmObjects	GB-小时数	每小时	在 30 天最低承诺结束之前，从 S3 Standard-IA 中删除的小对象 (小于 128KB) 的按比例存储使用量[3]
region-EarlyDelete-ZIA	GB-小时数	每小时	在 30 天最低承诺结束之前，从 S3 One Zone-IA 中删除的对象的按比例存储使用量[3]
region-EarlyDelete-ZIA-SmObjects	GB-小时数	每小时	在 30 天最低承诺结束之前，从 S3 One Zone-IA 中删除的小对象 (小于 128KB) 的按比例存储使用量[3]
region-Expedited-Retrieval-Bytes	GB	每小时	使用 Expedited S3 Glacier Flexible Retrieval 的数据量
region-Inventory-Objects Listed	对象	每小时	为采用清单列表的对象组（对象按存储桶或前缀分组）列出的对象数
region-Monitoring-Automation-INT	对象	每小时	S3 Intelligent-Tiering 存储类中被监控和自动分层的唯一对象的数量
region-MRAP-Out-Bytes	GB	每小时	通过 S3 多区域接入点端点，从区域中的桶中传出的数据量（MRAP 数据路由定价）。

Usage Type	单位	粒度	描述
region-MRAP-In-Bytes	GB	每小时	通过 S3 多区域接入点端点，从区域中的桶中传出的数据量（MRAP 数据路由定价）。
regiongroup1-regiongroup2--MRAP-Out-Bytes	GB	每小时	通过 S3 多区域接入点端点，从 *regiongroup1* 中的桶传输到位于 AWS 网络外部的 *regiongroup2* 中客户端的数据量。
regiongroup1-regiongroup2--MRAP-In-Bytes	GB	每小时	通过 S3 多区域接入点端点，从位于 AWS 网络外部的 *regiongroup2* 中的客户端传输到 *regiongroup1* 中桶的数据量。
region-OverwriteBytes-Copy-GDA	GB	每月	CopyObject 操作从 S3 Glacier Deep Archive 存储中覆盖的数据量
region-OverwriteBytes-Copy-GIR	GB	每月	CopyObject 操作从 S3 Glacier Instant Retrieval 存储中覆盖的数据量。
region-OverwriteBytes-Copy-GLACIER	GB	每月	CopyObject 操作从 S3 Glacier Flexible Retrieval 存储中覆盖的数据量
region-OverwriteBytes-Copy-INT	GB	每月	CopyObject 操作从 S3 Intelligent-Tiering 存储中覆盖的数据量

Usage Type	单位	粒度	描述
region-OverwriteBytes-Copy-RRS	GB	每月	CopyObject 操作从低冗余存储（RRS）存储中覆盖的数据量
region-OverwriteBytes-Copy-SIA	GB	每月	CopyObject 操作从 S3 Standard-IA 存储中覆盖的数据量
region-OverwriteBytes-Copy-STANDARD	GB	每月	CopyObject 操作从 S3 Standard 存储中覆盖的数据量
region-OverwriteBytes-Copy-ZIA	GB	每月	CopyObject 操作从 S3 One Zone-IA 存储中覆盖的数据量
region-OverwriteBytes-Put-GDA	GB	每月	PutObject 操作从 S3 Glacier Deep Archive 存储中覆盖的数据量
region-OverwriteBytes-Put-GIR	GB	每月	PutObject 操作从 S3 Glacier Instant Retrieval 存储中覆盖的数据量。
region-OverwriteBytes-Put-GLACIER	GB	每月	PutObject 操作从 S3 Glacier Flexible Retrieval 存储中覆盖的数据量
region-OverwriteBytes-Put-INT	GB	每月	PutObject 操作从 S3 Intelligent-Tiering 存储中覆盖的数据量
region-OverwriteBytes-Put-RRS	GB	每月	PutObject 操作从低冗余存储（RRS）存储中覆盖的数据量

Usage Type	单位	粒度	描述
region-OverwriteBytes-Put-SIA	GB	每月	PutObject 操作从 S3 Standard-IA 存储中覆盖的数据量
region-OverwriteBytes-Put-STANDARD	GB	每月	PutObject 操作从 S3 Standard 存储中覆盖的数据量
region-OverwriteBytes-Put-ZIA	GB	每月	PutObject 操作从 S3 One Zone-IA 存储中覆盖的数据量
region1-*region2*-S3RTC-In-Bytes	GB	每月	由 PutObject ReplTime 、GetObject ReplTime 、InitiateMultipartUploadReplTime 、UploadPartReplTime 、CompleteMultipartUploadReplTime 和 WriteACLReplTime 操作针对 S3 Replication Time Control（S3 RTC）从 *region2* 传输到 *region1* 的数据量

Usage Type	单位	粒度	描述
region1-*region2*-S3RTC-Out-Bytes	GB	每月	由 PutObject ReplTime 、GetObject ReplTime 、InitiateMultipartUploadReplTime 、UploadPartTReplTime 、CompleteMultipartUploadReplTime 和 WriteACLReplTime 操作针对 S3 Replication Time Control（S3 RTC）从 *region1* 传输到 *region2* 的数据量
region-Requests-GDA-Tier1	计数	每小时	对 S3 Glacier Deep Archive 对象的 PUT、COPY、POST、CreateMultipartUpload 、UploadPart 或 CompleteMultipartUpload 请求的数量[6]
region-Requests-GDA-Tier2	计数	每小时	对 S3 Glacier Deep Archive 对象的 GET 和 HEAD 请求的数量
region-Requests-GDA-Tier3	计数	每小时	S3 Glacier Deep Archive 标准还原请求的数量
region-Requests-GDA-Tier5	计数	每小时	批量 S3 Glacier Deep Archive 还原请求的数量

Usage Type	单位	粒度	描述
region-Requests-GIR-Tier1	计数	每小时	对 S3 Glacier Instant Retrieval 对象的 PUT、COPY 或 POST 请求的数量。
region-Requests-GIR-Tier2	计数	每小时	对 S3 Glacier Instant Retrieval 对象的 GET 和所有其它非 S3 Glacier Instant Retrieval-Tier1 请求的数量。
region-Requests-GLACIER-Tier1	计数	每小时	对 S3 Glacier Flexible Retrieval 对象的 PUT、COPY、POST、CreateMultipartUpload 、UploadPart 或 CompleteMultipartUpload 请求的数量 [6]
region-Requests-GLACIER-Tier2	计数	每小时	对 S3 Glacier Flexible Retrieval 对象的 GET 和所有其它未列出的请求的数量
region-Requests-INT-Tier1	计数	每小时	对 S3 Intelligent-Tiering 对象的 PUT、COPY 或 POST 请求的数量
region-Requests-INT-Tier2	计数	每小时	对 S3 Intelligent-Tiering 对象的 GET 和所有其它非 Tier1 请求的数量
region-Requests-SIA-Tier1	计数	每小时	对 S3 Standard-IA 对象的 PUT、COPY 或 POST 请求的数量

Usage Type	单位	粒度	描述
region-Requests-SIA-Tier2	计数	每小时	对 S3 Standard-IA 对象的 GET 和所有其它非 S3 Glacier Instant Retrieval-Tier1 请求的数量。
region-Requests-Tier1	计数	每小时	对 S3 Standard、RRS 和标签的 PUT、COPY 或 POST 请求以及对所有桶和对象的 LIST 请求的数量
region-Requests-Tier2	计数	每小时	GET 和其它非 Tier1 请求的数量
region-Requests-Tier3	计数	每小时	S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 的生命周期请求和标准 S3 Glacier Flexible Retrieval 还原请求的数量
region-Requests-Tier4	计数	每小时	转换到 S3 Glacier Instant Retrieval、S3 Intelligent-Tiering、S3 Standard-IA 或 S3 One Zone-IA storage 存储的生命周期转换数
region-Requests-Tier5	计数	每小时	批量 S3 Glacier Flexible Retrieval 还原请求的数量
region-Requests-Tier6	计数	每小时	批量 Expedited S3 Glacier Flexible Retrieval 还原请求的数量
region-Requests-Tier8	计数	每小时	S3 访问授权请求的数量

Usage Type	单位	粒度	描述
region-Requests-XZ-Tier1	计数	每小时	对 S3 Express One Zone 对象的 PUT 或 COPY 请求的数量
region-Requests-XZ-Tier2	计数	每小时	对 S3 Express One Zone 对象的 GET 和其它所有非 S3 Express One Zone-Tier1 请求的数量
region-Requests-ZIA-Tier1	计数	每小时	对 S3 One Zone-IA 对象的 PUT、COPY 或 POST 请求的数量
region-Requests-ZIA-Tier2	计数	每小时	对 S3 One Zone-IA 对象的 GET 和所有其它非 S3 One Zone-IA-Tier1 请求的数量
region-Retrieval-GIR	GB	每小时	从 S3 Glacier Instant Retrieval 存储检索的数据量。
region-Retrieval-SIA	GB	每小时	从 S3 Standard-IA 存储检索的数据量
region-Retrieval-XZ	GB	每小时	使用 S3 Express One Zone 存储的给定检索请求（PUT 或 COPY）中超过 512KB 的数据部分
region-Retrieval-ZIA	GB	每小时	从 S3 One Zone-IA 存储检索的数据量
region-S3DSSE-In-Bytes	GB	每月	Amazon S3 双重加密的数据量

Usage Type	单位	粒度	描述
region-S3DSSE-Out-Bytes	GB	每月	Amazon S3 解密的双重加密数据量
region-S3G-DataTransfer-In-Bytes	GB	每小时	为了从 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储中还原对象而传入 Amazon S3 的数据量
region-S3G-DataTransfer-Out-Bytes	GB	每小时	为了将对象转移到 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储而从 Amazon S3 传输的数据量
region-Select-Returned-Bytes	GB	每小时	使用 Select 请求从 S3 Standard存储返回的数据量
region-Select-Returned-GIR-Bytes	GB	每小时	通过 S3 Glacier Instant Retrieval 存储的选择请求返回的数据量。
region-Select-Returned-INT-Bytes	GB	每小时	使用 Select 请求从 S3 Intelligent-Tiering存储返回的数据量
region-Select-Returned-SIA-Bytes	GB	每小时	使用 Select 请求从 S3 Standard-IA 存储返回的数据量
region-Select-Returned-ZIA-Bytes	GB	每小时	使用 Select 请求从 S3 One Zone-IA 存储返回的数据量

Usage Type	单位	粒度	描述
region-Select-Scanned-Bytes	GB	每小时	使用 Select 请求从 S3 Standard存储扫描的数据量
region-Select-Scanned-GIR-Bytes	GB	每小时	通过 S3 Glacier Instant Retrieval 存储的选择请求扫描的数据量。
region-Select-Scanned-INT-Bytes	GB	每小时	使用 Select 请求从 S3 Intelligent-Tiering存储扫描的数据量
region-Select-Scanned-SIA-Bytes	GB	每小时	使用 Select 请求从 S3 Standard-IA 存储扫描的数据量
region-Select-Scanned-ZIA-Bytes	GB	每小时	使用 Select 请求从 S3 One Zone-IA 存储扫描的数据量
region-Standard-Retrieval-Bytes	GB	每小时	使用标准 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 请求检索的数据量
region-StorageAnalytics-ObjCount	对象	每小时	在每个存储类分析配置中监视的唯一对象的数量。
region-StorageLens-ObjCount	对象	每天	每个 S3 Storage Lens 存储统计管理工具控制面板中由 S3 Storage Lens 存储统计管理工具高级指标和建议跟踪的唯一对象的数量。

Usage Type	单位	粒度	描述
region-StorageLensFreeTier-ObjCount	对象	每天	每个 S3 Storage Lens 存储统计管理工具控制面板中由 S3 Storage Lens 存储统计管理工具使用量指标跟踪的唯一对象的数量。
StorageObjectCount	计数	每天	在给定存储桶中存储的对象数
region-TagStorage-TagHrs	标签小时数	每天	按小时报告的存储桶中所有对象上的标签总数
region-TimedStorage-ByteHrs	GB-月	每天	数据存储在 S3 Standard 存储中的 GB-月数
region-TimedStorage-GDA-ByteHrs	GB-月	每天	数据存储在 S3 Glacier Deep Archive 存储中的 GB-月数
region-TimedStorage-GDA-Staging	GB-月	每天	数据存储在 S3 Glacier Deep Archive 暂存存储中的 GB-月数
region-TimedStorage-GIR-ByteHrs	GB-月	每天	数据存储在 S3 Glacier Instant Retrieval 存储中的 GB-月数。
region-TimedStorage-GIR-SmObjects	GB-月	每天	小对象（小于 128 KB）存储在 S3 Glacier Instant Retrieval 存储中的 GB-月数。
region-TimedStorage-GlacierByteHrs	GB-月	每天	数据存储在 S3 Glacier Flexible Retrieval 存储中的 GB-月数

Usage Type	单位	粒度	描述
region-TimedStorage-GlacierStaging	GB-月	每天	数据存储在 S3 Glacier Flexible Retrieval 暂存存储中的 GB-月数
region-TimedStorage-INT-FA-ByteHrs	GB-月	每天	数据存储在 S3 Intelligent-Tiering 存储的频繁访问层中的 GB-月数 [5]
region-TimedStorage-INT-IA-ByteHrs	GB-月	每天	数据存储在 S3 Intelligent-Tiering 存储的不频繁访问层中的 GB-月数
region-TimedStorage-INT-AA-ByteHrs	GB-月	每天	数据存储在 S3 Intelligent-Tiering 存储的归档访问层中的 GB-月数
region-TimedStorage-INT-AIA-ByteHrs	GB-月	每天	数据存储在 S3 Intelligent-Tiering 存储的归档即时访问层中的 GB-月数
region-TimedStorage-INT-DAA-ByteHrs	GB-月	每天	数据存储在 S3 Intelligent-Tiering 存储的深层归档访问层中的 GB-月数
region-TimedStorage-RRS-ByteHrs	GB-月	每天	数据存储在低冗余存储（RRS）存储中的 GB-月数
region-TimedStorage-SIA-ByteHrs	GB-月	每天	数据存储在 S3 Standard-IA 存储中的 GB-月数
region-TimedStorage-SIA-SmObjects	GB-月	每天	小对象（小于 128 KB）存储在 S3 Standard-IA 存储中的 GB-月数 [4]

Usage Type	单位	粒度	描述
region-TimedStorage-XZ-ByteHrs	GB-月	每天	数据存储在 S3 Express One Zone 存储中的 GB-月数
region-TimedStorage-ZIA-ByteHrs	GB-月	每天	数据存储在 S3 One Zone-IA 存储中的 GB-月数
region-TimedStorage-ZIA-SmObjects	GB-月	每天	小对象（小于 128 KB）存储在 S3 One Zone-IA 存储中的 GB-月数
region-Upload-XZ	GB	每小时	使用 S3 Express One Zone 的给定上传请求（PUT 或 COPY）中超过 512KB 的数据量

注意

1. 如果您在传输完成之前终止传输，传输的数据量可能会超过您的应用程序接收的数据量。由于无法立即执行传输终止请求，并且有些数据量可能在传输中，正在等待终止请求的执行，因此会出现此差异。传输中数据以传“出”数据计费。
2. 当归档到 S3 Glacier Instant Retrieval、S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类的对象在最小存储承诺（对于 S3 Glacier Instant Retrieval 和 S3 Glacier Flexible Retrieval 为 90 天，或对于 S3 Glacier Deep Archive 为 180 天）过去之前被删除、覆盖或转换为不同的存储类时，对剩余的天数会以 GB 为单位按比例收费。
3. 对于位于 S3 Standard-IA 或 S3 One Zone-IA 存储中的对象，当它们在 30 天之前被删除、覆盖或转换为其它存储类时，对剩余的天数会以 GB 为单位按比例收费。
4. 对于位于 S3 Standard-IA 或 S3 One Zone-IA 存储中的小对象（小于 128KB），当它们在 30 天之前被删除、覆盖或转换为其它存储类时，对剩余的天数会以 GB 为单位按比例收费。
5. S3 Intelligent-Tiering 存储类中的对象没有最小可计费对象大小。小于 128 KB 的对象不受监控，也不符合自动分层条件。较小的对象始终存储在 S3 Intelligent-Tiering 频繁访问层中。
6. 当您向 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类发起 `CreateMultipartUpload`、`UploadPart` 或 `UploadPartCopy` 请求时，请求将按 S3 Standard 请求费率计费，直到您完成分段上传。完成上传后，单个 `CompleteMultipartUpload` 请求将按

目标 S3 Glacier 存储的 PUT 费率计费。在上传完成之前，针对 S3 Glacier Flexible Retrieval 存储类的 PUT 的正在上传的分段采用 S3 Standard 存储费率，按 S3 Glacier Flexible Retrieval 暂存存储计费。同样，在上传完成之前，针对 S3 Glacier Deep Archive 存储类的 PUT 的正在上传的分段采用 S3 Standard 存储费率，按 S3 Glacier Deep Archive 暂存存储计费。

7. 对于大小不超过 512KB 的请求，S3 Express One Zone 按请求收取统一费用。对于请求中超过 512KB 的部分，PUT 请求和 GET 请求将按每 GB 额外收费。
8. 有关 S3 Express One Zone 存储类支持的功能的信息，请参阅 S3 Express One Zone 不支持的 Amazon S3 功能。
9. 单位以 GB 计费的使用量类型在使用量报告中以字节为单位进行计算。
10. GB-月的计算方法是：取 GB 小时总数，将一个月内的这些数量汇总，然后除以该月的小时数。要了解更多信息，请参阅常见问题：使用 Amazon S3 是如何收费和计费的？

> **Note**
>
> 通常，S3 存储桶拥有者需要为获得 HTTP 200 OK 成功响应和 HTTP 4XX 客户端错误响应的请求付费。存储桶拥有者无需为 HTTP 5XX 服务器错误响应（例如 HTTP 503 Slow Down 错误）付费。有关不予计费的 HTTP 3XX 和 4XX 状态码下的 S3 错误代码的更多信息，请参阅 Amazon S3 错误响应的账单计费。有关在存储桶配置为申请方付款存储桶时的账单费用的更多信息，请参阅申请方付款的费用支付方式。

跟踪使用率报告中的操作

操作描述指定的使用类型对您的 AWS 对象或存储桶所采取的操作。操作由不言自明的代码表示，如 PutObject 或 ListBucket。要查看存储桶上的哪些操作生成了特定类型的用途，请使用这些代码。当您创建使用率报告时，可以选择包括 All Operations 或特定操作 (例如 GetObject) 以进行报告。

更多信息

- AWSAmazon S3 的使用情况报告
- Amazon S3 的 AWS Billing报告
- Amazon S3 定价
- Amazon S3 常见问题

Amazon S3 错误响应的账单计费

通常，S3 存储桶拥有者需要为获得 HTTP 200 OK 成功响应和 HTTP 4XX 客户端错误响应的请求付费。存储桶拥有者无需为 HTTP 5XX 服务器错误响应（例如 HTTP 503 Slow Down 错误）付费。有关在存储桶配置为申请方付款存储桶时的账单费用的更多信息，请参阅申请方付款的费用支付方式。

下表列出了不予计费的 HTTP 3XX 和 4XX 状态码下的特定错误代码。对于配置了网站托管的存储桶，当 S3 返回自定义错误文档或进行自定义重定向时，仍将收取适用的请求和其它费用。

> **Note**
>
> 对于 AccessDenied（HTTP 403 Forbidden），如果请求是在存储桶拥有者的个人 AWS 账户或 AWS 组织外部发起，则 S3 不会向存储桶拥有者收费。

HTTP 状态代码	错误代码	错误代码说明	
301 永久移动	PermanentRedirect	必须使用指定的端点来寻址您正在尝试访问的存储桶。请将未来的所有请求发送到此端点。	
	PermanentRedirectControlError	必须使用指定的端点来寻址您正在尝试访问的 API 操作。请将未来的所有请求发送到此端点。	
307 临时重新导向	TemporaryRedirect	域名系统（DNS）服务器更新时您会被重定向至存储桶。	
400 错误请求	AuthorizationHeaderMalformed	您提供的授权标头无效。	
	AuthorizationQueryParametersError	您提供的授权查询参数无效。	

HTTP 状态代码	错误代码	错误代码说明	
	ConnectionClosedByRequester	如果在读取 WriteGetObjectResponse 正文时遇到错误，则返回给原始调用方。	
	DeviceNotActiveError	设备当前处于不活动状态。	
	EndpointNotFound	将请求定向到正确的端点。	
	ExpiredToken	提供的令牌过期。	
	IllegalLocationConstraintException	您正在尝试从存储桶所在区域以外的区域访问存储桶。要避免此错误，请使用 `--region` 选项。例如：`aws s3 cp awsexample.txt s3://amzn-s3-demo-bucket/ --region ap-east-1` 。	
	InvalidArgument	出现此错误可能的原因如下： • 指定的参数无效。 • 请求缺少必需的标头。 • 指定的参数不完整或格式有误。 • 指定的参数长度必须大于或等于 3。	

HTTP 状态代码	错误代码	错误代码说明	
	InvalidBucketOwnerAWSAccountID	预期的存储桶拥有者参数的值必须为 AWS 账户 ID。	
	InvalidDigest	指定的 Content-MD5 或校验和值无效。	
	InvalidEncryptionAlgorithmError	指定的加密请求无效。有效值为 AES256。	
	InvalidHostHeader	请求中提供的主机标头使用了不正确的样式寻址。	
	InvalidHttpMethod	请求是使用意外的 HTTP 方法发出的。	

HTTP 状态代码	错误代码	错误代码说明	
	InvalidRequest	出现此错误可能的原因如下： • 请求使用了错误的签名版本。请使用 AWS4-HMAC-SHA256 （签名版本 4）。 • 只能为现有存储桶创建接入点。 • 接入点未处于可以删除的状态。 • 只能列出现有存储桶的接入点。 • 下一个令牌无效。 • 生命周期规则中必须至少指定一个操作。 • 必须至少指定一个生命周期规则。 • 生命周期规则的数量不得超过所允许的 1000 条规则的限制。 • MaxResults 参数的范围无效。 • SOAP 请求必须通过 HTTPS 连接发出。	

HTTP 状态代码	错误代码	错误代码说明	
		• 名称不符合 DNS 的存储桶不支持 Amazon S3 Transfer Acceleration。 • 名称中包含点（.）的存储桶不支持 Amazon S3 Transfer Acceleration。 • Amazon S3 Transfer Acceleration 端点仅支持虚拟样式的请求。 • 此存储桶上未配置 Amazon S3 Transfer Acceleration。 • 此存储桶上已禁用 Amazon S3 Transfer Acceleration。 • 该存储桶上不支持 Amazon S3 Transfer Acceleration。如需帮助，请联系 AWS Support。 • 此存储桶上不能启用 Amazon S3 Transfer Acceleration。如需帮助，请联系 AWS Support。 • HTTP 标头和查询参数中提供的值相互冲突。	

HTTP 状态代码	错误代码	错误代码说明	
		• HTTP 标头和 POST 表单字段中提供的值相互冲突。 • 在大于 5GB 的对象上发出 CopyObject 请求。	
	InvalidSessionException	如果会话因超时或过期而不再存在，则返回。	
	InvalidSignature	服务器计算出的请求签名与您提供的签名不匹配。请检查您的 AWS 秘密访问密钥和签名方法。有关更多信息，请参阅签署和对 REST 请求进行身份验证。	
	InvalidSOAPRequest	SOAP 请求正文无效。	
	InvalidStorageClass	指定的存储类无效。	
	InvalidTag	您的请求包含无效的标签输入。例如，您的请求可能包含重复的键、过长的键或值，或包含系统标签。	
	InvalidToken	提供的令牌格式不正确或者无效。	
	InvalidURI	无法解析指定的 URI。	
	KeyTooLongError	键过长。	

HTTP 状态代码	错误代码	错误代码说明	
	KMS.DisabledException	由于指定的 KMS 密钥未启用，请求被拒绝。	
	KMS.InvalidKeyUsageException	由于以下原因之一，请求被拒绝： • KMS 密钥的 KeyUsage 值与 API 操作不兼容。 • 为操作指定的加密算法或签名算法与 KMS 密钥（KeySpec）中的密钥材料类型不兼容。 要加密、解密、重新加密和生成数据密钥，KeyUsage 必须为 ENCRYPT_DECRYPT。要对消息进行签名和验证，KeyUsage 必须为 SIGN_VERIFY。要生成和验证消息身份验证代码（MAC），KeyUsage 必须为 GENERATE_VERIFY_MAC。要获取密钥协议密钥，KeyUsage 必须为 KEY_AGREEMENT。要查找 KMS 密钥的 KeyUsage，请使用 DescribeKey 操作。要查找特定 KMS 密钥支持的加密或签名算法，请使用 DescribeKey 操作。	

HTTP 状态代码	错误代码	错误代码说明	
	KMS.KMSInvalidStateException	由于指定资源的状态对此请求无效，请求被拒绝。此异常意味着以下情况之一： • KMS 密钥的密钥状态与操作不兼容。 要查找密钥状态，请使用 DescribeKey 操作。有关哪些密钥状态与每个 KMS 操作兼容的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 Key states of AWS KMS keys。 • 对于自定义密钥存储中 KMS 密钥的加密操作，此异常表示存在一般性故障，原因可能有很多。要确定原因，请参阅随异常显示的错误消息。	
	KMS.NotFoundException	由于找不到指定的实体或资源，请求被拒绝。	

HTTP 状态代码	错误代码	错误代码说明	
	LambdaInvalidResponse	当 WriteGetObjectResponse 向 AWS Lambda 响应 ValidationError 时，返回给原始调用方。有关更多详细信息，请参阅 ValidationError 消息。并非所有 ValidationError 情况都会导致 LambdaInvalidResponse 错误。	
	LambdaInvocationFailed	Lambda 函数调用失败。当 S3 对象 Lambda 无法成功调用已配置的 Lambda 函数时，调用方可能会收到以下错误。错误消息可能包含有关在调用函数时 AWS Lambda 服务返回的最终错误的详细信息（例如，状态代码、错误代码、错误消息和请求 ID）。	
	MalformedACLError	您提供的 ACL 格式有误，或者没有根据我们发布的架构进行验证。	
	MalformedPOSTRequest	您的 POST 请求正文的分段/表单数据格式不正确。	
	MalformedXML	您提供的 XML 格式不正确，或者没有根据我们发布的架构进行验证。	
	MaxPostPreDataLengthExceededError	上传文件前的 POST 请求字段太大。	

HTTP 状态代码	错误代码	错误代码说明	
	MetadataTooLarge	您的元数据标头超过了允许的最大元数据大小。	
	MissingAttachment	预计会有 SOAP 附件，但未找到任何附件。	
	MissingRequestBodyError	您发送了一个空的 XML 文档作为请求。	
	MissingSecurityHeader	请求缺少必需的标头。	
	NoLoggingStatusForKey	不存在针对某个键的日志记录状态子资源之类的内容。	
	NotDeviceOwnerError	生成令牌的设备不归经过身份验证的用户拥有。	
	ResponseInterrupted	如果在读取 WriteGetObjectResponse 正文时遇到错误，则返回给原始调用方。	
	RequestHeaderSectionTooLarge	用于发出请求的请求标头和查询参数超出允许的最大大小	
	TokenCodeInvalidError	您提供的序列号和/或令牌代码无效。	
	UnexpectedContent	此请求包含不支持的内容。	
	UnsupportedArgument	该请求包含不支持的参数。	

HTTP 状态代码	错误代码	错误代码说明	
	UnsupportedSignature	所提供的请求使用不受支持的 STS 令牌版本进行签名，或者签名版本不受支持。	
	UserKeyMustBeSpecified	存储桶 POST 请求必须包含指定的字段名称。如果已指定，请检查字段的顺序。	
	IncorrectEndpoint	指定的存储桶存在于其它区域中。将请求定向到正确的端点。	
	ValidationError	验证错误可能从 WriteGetObjectResponse API 操作中返回，并且可能由于多种原因而发生。有关更多详细信息，请参阅错误消息。	
403 禁止访问	RequestTimeTooSkewed	请求时间和服务器时间之间的差异太大。	
	SignatureDoesNotMatch	服务器计算出的请求签名与您提供的签名不匹配。请检查您的 AWS 秘密访问密钥和签名方法。有关更多信息，请参阅 REST 身份验证和 SOAP 身份验证。	

HTTP 状态代码	错误代码	错误代码说明	
	NotSignedUp	您的账户未注册 Amazon S3 服务。您必须先注册，然后才能使用 Amazon S3。您可以通过以下 URL 注册：https://aws.amazon.com/s3	
	InvalidSecurity	提供的安全凭证无效。	
	InvalidPayer	已禁用针对此对象的所有访问权限。如需更多帮助，请参阅联系我们。	
	InvalidAccessKeyId	您提供的 AWS 访问密钥 ID 在记录中不存在。	
	AccountProblem	您的 AWS 账户存在问题，导致无法成功完成操作。如需更多帮助，请参阅联系我们。	
	UnauthorizedAccessError	仅适用于中国区域。在向没有 ICP 许可证的存储桶发出请求时返回。有关更多信息，请参阅 ICP Recordal。	
	UnexpectedIPError	仅适用于中国区域。由于 IP 不符合预期，请求被拒绝。	
	MissingAuthenticationToken	未对请求进行签名。	

HTTP 状态代码	错误代码	错误代码说明	
	LambdaPermissionError	调用方未获授权，无法调用 Lambda 函数。调用方必须具有调用 Lambda 函数的权限。检查附加到调用方的策略，并确保这些策略已获准将 `lambda:Invoke` 用于所配置的函数。错误消息可能包含有关在调用函数时 Lambda 服务返回的最终错误的详细信息（例如，状态代码、错误代码、错误消息和请求 ID）。	
404 未找到	LambdaNotFound	找不到 AWS Lambda 函数。尝试调用配置的 Lambda 函数、版本或别名时找不到它。确保 S3 对象 Lambda 接入点配置指向正确的 Lambda 函数 ARN。错误消息可能包含有关在调用函数时 AWS Lambda 服务返回的最终错误的详细信息（例如，状态代码、错误代码、错误消息和请求 ID）。	
	NoSuchAsyncRequest	找不到指定的请求。	
	NoSuchObjectLockConfiguration	指定的对象没有 ObjectLock 配置。	

HTTP 状态代码	错误代码	错误代码说明	
	NoSuchUpload	指定的分段上传不存在。上传 ID 可能无效，或者分段上传可能已中止或已完成。	
	NoSuchWebsiteConfiguration	指定的存储桶没有网站配置。	
	NoTransformationDefined	找不到此对象 Lambda 接入点的转换。	
	ObjectLockConfigurationNotFoundError	此存储桶不存在对象锁定配置。	
405 不允许的方法	MethodNotAllowed	不允许对此资源使用指定的方法。	
409 冲突	BucketAlreadyExists	请求的存储桶名称不可用。存储桶命名空间由系统的所有用户共享。请指定其它名称，然后重试。	
	InvalidBucketState	请求对于存储桶的当前状态无效。	
	OperationAborted	当前正在对此资源执行冲突的条件操作。请重试。	
411 需要长度	MissingContentLength	您必须提供 Content-Length HTTP 标头。	
412 前提条件失败	RequestIsNotMultiPartContent	存储桶 POST 请求必须是附件类型的分段/表单数据。	

HTTP 状态代码	错误代码	错误代码说明	
416 请求的范围无法满足	InvalidRange	请求的范围对请求无效。尝试其它范围。	

使用 Amazon S3 Select 筛选和检索数据

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

利用 Amazon S3 Select，您可以使用结构化查询语言（SQL）语句筛选 Amazon S3 对象的内容，并仅检索所需的部分数据。通过使用 Amazon S3 Select 筛选此数据，您可以减少 Amazon S3 传输的数据量，这将减少检索此数据所需的成本和延迟。

Amazon S3 Select 只允许您一次查询一个对象。它适用于以 CSV、JSON 或 Apache Parquet 格式存储的对象。它还使用通过 GZIP 或 BZIP2 压缩的对象（仅对于 CSV 和 JSON 对象）和服务器端加密的对象。可以将结果的格式指定为 CSV 或 JSON，并且可以确定结果中记录的分隔方式。

可以在请求中将 SQL 表达式传递给 Amazon S3。Amazon S3 Select 支持一部分 SQL。有关 Amazon S3 Select 支持的 SQL 元素的更多信息，请参阅 Amazon S3 Select 的 SQL 参考。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）、`SelectObjectContent` REST API 操作或 AWS SDK 执行 SQL 查询。

> **Note**
>
> Amazon S3 控制台将返回的数据量限定为 40 MB。要检索更多数据，请使用 AWS CLI 或 API。

要求和限制

以下是使用 Amazon S3 Select 的要求：

- 您必须拥有所查询的对象的 `s3:GetObject` 权限。
- 如果您查询的对象已使用具有客户提供的加密密钥的客户端加密（SSE-C）进行加密，则必须使用 `https`，并且您必须在请求中提供加密密钥。

使用 Amazon S3 Select 时存在以下限制：

- S3 Select 对于每个请求只能查询一个对象。
- SQL 表达式的最大长度为 256 KB。
- 输入或结果中记录的最大长度为 1 MB。
- Amazon S3 Select 只能使用 JSON 输出格式发出嵌套数据。
- 您无法查询存储在 S3 Glacier Flexible Retrieval、S3 Glacier Deep Archive 或低冗余存储（RRS）存储类中的对象。您无法查询存储在 S3 Intelligent-Tiering 存档访问层或 S3 Intelligent-Tiering 深度存档访问层中的对象。有关存储类的更多信息，请参阅 使用 Amazon S3 存储类。

在将 Amazon S3 Select 用于 Parquet 对象时，以下额外限制适用：

- Amazon S3 Select 仅支持使用 GZIP 或 Snappy 的列式压缩。Amazon S3 Select 对于 Parquet 对象不支持整个对象压缩。
- Amazon S3 Select 不支持 Parquet 输出。您必须将输出格式指定为 CSV 或 JSON。
- 未压缩的最大行组大小为 512MB。
- 您必须使用在对象的模式中指定的数据类型。
- 选择重复字段将只返回最后一个值。

构建请求

在构造请求时，您提供通过使用 `InputSerialization` 对象查询的对象的详细信息。您提供要使用 `OutputSerialization` 对象返回结果的方式的详细信息。您还包含 Amazon S3 用于筛选请求的 SQL 表达式。

有关构造 Amazon S3 Select 请求的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 SelectObjectContent。您也可以参阅以下部分中的某个开发工具包代码示例。

使用扫描范围的请求

借助 Amazon S3 Select，您可以通过指定要查询的字节范围来扫描对象的子集。利用此功能，您可以将工作分为一系列非重叠扫描范围的独立 Amazon S3 Select 请求，从而并行扫描整个对象。

扫描范围不需要与记录边界对齐。Amazon S3 Select 扫描范围请求在您指定的字节范围内运行。查询将处理在指定的扫描范围内开始但延伸到该扫描范围之外的记录。例如，下面显示了包含一系列记录（采用以行分隔的 CSV 格式）的 Amazon S3 对象：

```
A,B
C,D
D,E
E,F
G,H
I,J
```

假设您使用 Amazon S3 Select `ScanRange` 参数，并从（字节）1 开始，在（字节）4 结束。因此，扫描范围将从“,”开始，并扫描直到记录结尾 `C`。您的扫描范围请求将返回结果 `C, D`，因为这是记录的结尾。

Amazon S3 Select 扫描范围请求支持 Parquet、CSV（不带引号的分隔符）或 JSON 对象（仅在 `LINES` 模式下）。CSV 和 JSON 对象必须未压缩。对于基于行的 CSV 和 JSON 对象，当将扫描范围指定为 Amazon S3 Select 请求的一部分时，将处理在该扫描范围内开始的所有记录。对于 Parquet 对象，将处理在请求的扫描范围内开始的所有行组。

Amazon S3 Select 扫描范围请求可在 AWS CLI、Amazon S3 API 和 AWS SDK 中使用。您可以在 Amazon S3 Select 请求中使用 `ScanRange` 参数来实现此功能。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 SelectObjectContent。

错误

如果在尝试运行查询时遇到问题，Amazon S3 Select 将返回错误代码和关联的错误消息。有关错误代码和描述的列表，请参阅 Amazon Simple Storage Service API 参考中的错误响应页面的 SELECT 对象内容错误代码列表部分。

有关 Amazon S3 Select 的更多信息，请参阅以下主题。

主题

- 在对象上使用 Amazon S3 Select 的示例
- Amazon S3 Select 的 SQL 参考

在对象上使用 Amazon S3 Select 的示例

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

您可以使用 S3 Select，通过 Amazon S3 控制台、REST API 和 AWS SDK 从一个对象中选择内容。

有关 S3 Select 支持的 SQL 函数的更多信息，请参阅SQL 函数。

使用 S3 控制台

在 Amazon S3 控制台中选择对象中的内容

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 选择包含您想要从中选择内容的对象的桶，然后选择对象的名称。
4. 选择对象操作，然后选择使用 S3 Select 进行查询。
5. 根据输入数据的格式配置输入设置。
6. 根据要接收的输出的格式配置输出设置。
7. 要从所选对象中提取记录，请在 SQL 查询下输入 SELECT SQL 命令。有关如何编写 SQL 命令的更多信息，请参阅Amazon S3 Select 的 SQL 参考。
8. 输入 SQL 查询后，选择运行 SQL 查询。然后，在查询结果下，您可以看到 SQL 查询的结果。

使用 REST API

您可以使用 AWS SDK 从对象中选择内容。然而，如果您的应用程序需要它，则可以直接发送 REST 请求。有关请求和响应格式的更多信息，请参阅 SelectObjectContent。

使用 AWS SDK

您可以使用 Amazon S3 Select 通过 `selectObjectContent` 方法选择对象的一些内容。如果此方法成功，它将返回 SQL 表达式的结果。

Java

以下 Java 代码返回对象 (包含以 CSV 格式存储的数据) 中存储的每条记录的第一列的值。它还请求返回 `Progress` 和 `Stats` 消息。必须提供有效的存储桶名称和包含 CSV 格式的数据的对象。

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
package com.amazonaws;

import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CSVInput;
import com.amazonaws.services.s3.model.CSVOutput;
import com.amazonaws.services.s3.model.CompressionType;
import com.amazonaws.services.s3.model.ExpressionType;
import com.amazonaws.services.s3.model.InputSerialization;
import com.amazonaws.services.s3.model.OutputSerialization;
import com.amazonaws.services.s3.model.SelectObjectContentEvent;
import com.amazonaws.services.s3.model.SelectObjectContentEventVisitor;
import com.amazonaws.services.s3.model.SelectObjectContentRequest;
import com.amazonaws.services.s3.model.SelectObjectContentResult;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.concurrent.atomic.AtomicBoolean;

import static com.amazonaws.util.IOUtils.copy;

/**
 * This example shows how to query data from S3Select and consume the response in
 the form of an
 * InputStream of records and write it to a file.
 */

public class RecordInputStreamExample {

    private static final String BUCKET_NAME = "${my-s3-bucket}";
    private static final String CSV_OBJECT_KEY = "${my-csv-object-key}";
    private static final String S3_SELECT_RESULTS_PATH = "${my-s3-select-results-
path}";
    private static final String QUERY = "select s._1 from S3Object s";
```

```
   public static void main(String[] args) throws Exception {
       final AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();

       SelectObjectContentRequest request = generateBaseCSVRequest(BUCKET_NAME,
CSV_OBJECT_KEY, QUERY);
       final AtomicBoolean isResultComplete = new AtomicBoolean(false);

       try (OutputStream fileOutputStream = new FileOutputStream(new File
(S3_SELECT_RESULTS_PATH));
            SelectObjectContentResult result =
s3Client.selectObjectContent(request)) {
           InputStream resultInputStream =
result.getPayload().getRecordsInputStream(
                   new SelectObjectContentEventVisitor() {
                       @Override
                       public void visit(SelectObjectContentEvent.StatsEvent event)
                       {
                           System.out.println(
                                   "Received Stats, Bytes Scanned: " +
event.getDetails().getBytesScanned()
                                           +  " Bytes Processed: " +
event.getDetails().getBytesProcessed());
                       }

                       /*
                        * An End Event informs that the request has finished
successfully.
                        */
                       @Override
                       public void visit(SelectObjectContentEvent.EndEvent event)
                       {
                           isResultComplete.set(true);
                           System.out.println("Received End Event. Result is
complete.");
                       }
                   }
           );

           copy(resultInputStream, fileOutputStream);
       }

       /*
        * The End Event indicates all matching records have been transmitted.
        * If the End Event is not received, the results may be incomplete.
```

```
             */
            if (!isResultComplete.get()) {
                throw new Exception("S3 Select request was incomplete as End Event was
     not received.");
            }
        }

        private static SelectObjectContentRequest generateBaseCSVRequest(String bucket,
     String key, String query) {
            SelectObjectContentRequest request = new SelectObjectContentRequest();
            request.setBucketName(bucket);
            request.setKey(key);
            request.setExpression(query);
            request.setExpressionType(ExpressionType.SQL);

            InputSerialization inputSerialization = new InputSerialization();
            inputSerialization.setCsv(new CSVInput());
            inputSerialization.setCompressionType(CompressionType.NONE);
            request.setInputSerialization(inputSerialization);

            OutputSerialization outputSerialization = new OutputSerialization();
            outputSerialization.setCsv(new CSVOutput());
            request.setOutputSerialization(outputSerialization);

            return request;
        }
    }
```

JavaScript

有关将 AWS SDK for JavaScript 与 S3 `SelectObjectContent` API 操作一起使用从 Amazon S3 中存储的 JSON 和 CSV 文件中选择记录的 JavaScript 示例，请参阅博客文章在 AWS SDK for JavaScript 中引入对 Amazon S3 Select 的支持。

Python

有关使用 SQL 查询来搜索通过使用 S3 Select 以逗号分隔值（CSV）文件形式加载到 Amazon S3 中的数据的 Python 示例，请参阅博客文章使用 Amazon S3 Select 在没有服务器或数据库的情况下查询数据。

Amazon S3 Select 的 SQL 参考

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

此参考包含 Amazon S3 Select 支持的结构化查询语言（SQL）元素的描述。

主题

SELECT 命令

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

Amazon S3 Select 仅支持 `SELECT` SQL 命令。`SELECT` 支持以下 ANSI 标准子句：

- `SELECT list`
- `FROM` 子句
- `WHERE` 子句
- `LIMIT` 子句

> **Note**
>
> Amazon S3 Select 查询目前不支持子查询或联接。

SELECT list

SELECT 列表指定希望查询返回的列、函数和表达式。列表表示查询的输出。

```
SELECT *
SELECT projection1 AS column_alias_1, projection2 AS column_alias_2
```

第一个带 *（星号）的 SELECT 形式按原样返回每个传递 WHERE 子句的行。第二个 SELECT 形式使用用户定义的输出标量表达式 ***projection1*** 和 ***projection2*** 为每列创建一行。

FROM 子句

Amazon S3 Select 支持以下形式的 FROM 子句：

```
FROM table_name
FROM table_name alias
FROM table_name AS alias
```

在 FROM 子句的每种形式中，table_name 都是被查询的 S3Object。对于来自传统关系数据库的用户，可以将其视为一个数据库架构，其中包含一个表的多个视图。

按照标准 SQL，FROM 子句将会创建在 WHERE 子句中筛选并在 SELECT 列表中投影的行。

对于在 Amazon S3 Select 中存储的 JSON 对象，您也可以使用以下形式的 FROM 子句：

```
FROM S3Object[*].path
FROM S3Object[*].path alias
FROM S3Object[*].path AS alias
```

使用这种形式的 FROM 子句，您可以从 JSON 对象内的数组或对象中进行选择。您可以使用以下形式之一指定 path：

- 通过名称（在对象中）：.*name* 或 ['*name*']
- 通过索引（在数组中）：[*index*]
- 通过通配符（在对象中）：.*

- 通过通配符（在数组中）：[*]

> **Note**
>
> - 这种形式的 FROM 子句只适用于 JSON 对象。
> - 通配符始终至少发出一条记录。如果没有匹配的记录，Amazon S3 Select 将发射值 MISSING。在输出序列化期间（在查询完成运行后），Amazon S3 Select 会将 MISSING 值替换为空记录。
> - 聚合函数（AVG、COUNT、MAX、MIN 和 SUM）会跳过 MISSING 值。
> - 如果您在使用通配符时未提供别名，则可使用路径中的最后一个元素引用行。例如，可以使用查询 SELECT price FROM S3Object[*].books[*].price 从书籍列表中选择所有价格。如果路径以通配符而不是名称结束，则可使用值 _1 来引用行。例如，您可以使用查询 SELECT price FROM S3Object[*].books[*].price，而不是 SELECT _1.price FROM S3Object[*].books[*]。
> - Amazon S3 Select 始终将 JSON 文档视为根级值组成的数组。因此，即使您查询的 JSON 对象只有一个根元素，FROM 子句也必须以 S3Object[*] 开头。但为了确保兼容性，Amazon S3 Select 允许您在不包含路径时省略通配符。因此，完整的子句 FROM S3Object 等效于 FROM S3Object[*] as S3Object。如果包含路径，则还必须使用通配符。因此，FROM S3Object 和 FROM S3Object[*].*path* 都是有效子句，但 FROM S3Object.*path* 不是。

Example

示例：

示例 1

此示例显示使用以下数据集和查询时的结果：

```
{ "Rules": [ {"id": "1"}, {"expr": "y > x"}, {"id": "2", "expr": "z = DEBUG"} ]}
{ "created": "June 27", "modified": "July 6" }
```

```
SELECT id FROM S3Object[*].Rules[*].id
```

```
{"id":"1"}
```

```
{}
{"id":"2"}
{}
```

Amazon S3 Select 因以下原因生成每个结果：

- `{"id":"id-1"}` – `S3Object[0].Rules[0].id` 产生了匹配。
- `{}` – `S3Object[0].Rules[1].id` 与记录不匹配，因此 Amazon S3 Select 发出 `MISSING`，后者随后在输出序列化期间变为空记录并返回。
- `{"id":"id-2"}` – `S3Object[0].Rules[2].id` 产生了匹配。
- `{}` – `S3Object[1]` 对于 `Rules` 不匹配，因此 Amazon S3 Select 发出 `MISSING`，后者随后在输出序列化期间变为空记录并返回。

如果您不希望 Amazon S3 Select 在找不到匹配项时返回空记录，则可测试值 `MISSING`。以下查询将返回与上一查询相同的结果，但会省略空值：

```
SELECT id FROM S3Object[*].Rules[*].id WHERE id IS NOT MISSING
```

```
{"id":"1"}
{"id":"2"}
```

示例 2

此示例显示使用以下数据集和查询时的结果：

```
{ "created": "936864000", "dir_name": "important_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": ".aws" }, { "name": "downloads" } ], "owner": "Amazon
 S3" }
{ "created": "936864000", "dir_name": "other_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": "my stuff" }, { "name": "backup" } ], "owner": "User" }
```

```
SELECT d.dir_name, d.files FROM S3Object[*] d
```

```
{"dir_name":"important_docs","files":[{"name":"."},{"name":".."},{"name":".aws"},
{"name":"downloads"}]}
{"dir_name":"other_docs","files":[{"name":"."},{"name":".."},{"name":"my stuff"},
{"name":"backup"}]}
```

```
SELECT _1.dir_name, _1.owner FROM S3Object[*]
```

```
{"dir_name":"important_docs","owner":"Amazon S3"}
{"dir_name":"other_docs","owner":"User"}
```

WHERE 子句

WHERE 子句遵循以下语法：

```
WHERE condition
```

WHERE 子句根据 *condition* 筛选行。condition 是具有布尔结果的表达式。只有 condition 计算结果为 TRUE 的行才会在结果中返回。

LIMIT 子句

LIMIT 子句遵循以下语法：

```
LIMIT number
```

LIMIT 子句根据 *number* 限制您希望查询返回的记录数。

属性访问

SELECT 和 WHERE 子句可以使用以下部分的任一方法引用记录数据，具体取决于将查询的文件是采用 CSV 还是 JSON 格式。

CSV

- 列编号 – 您可以引用列名为 *_N* 的行的第 N 列，其中 *N* 是列位置。位置计数从 1 开始。例如，第一列名为 _1，第二列名为 _2。

  您可以通过 *_N* 或 *alias*.*_N* 引用列。例如，_2 和 myAlias._2 都是有效的，都可用于引用 SELECT 列表中的列和 WHERE 子句。
- Column Headers (列标头) – 对于具有标头行的 CSV 格式的对象，可在 SELECT 列表和 WHERE 子句中使用这些标头。尤其在传统 SQL 的 SELECT 和 WHERE 子句表达式中，可以通过 *alias*.*column_name* 或 *column_name* 引用列。

JSON

- Document (文档) – 可将 JSON 文档字段作为 *alias*.*name* 进行访问。还可以访问嵌套字段，例如 *alias*.*name1*.*name2*.*name3*。
- 列表 – 可以将零基索引与 [] 运算符结合使用以访问 JSON 列表中的元素。例如，可以通过 *alias*[1] 访问列表中第二个元素。例如，您可以将访问列表元素与字段结合起来，例如 *alias*.*name1*.*name2*[1].*name3*。
- 示例：将此 JSON 对象视为示例数据集：

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
     {"project_name":"project1", "completed":false},
     {"project_name":"project2", "completed":true}
    ]
}
```

示例 1

以下查询返回以下结果：

```
Select s.name from S3Object s
```

```
{"name":"Susan Smith"}
```

示例 2

以下查询返回以下结果：

```
Select s.projects[0].project_name from S3Object s
```

```
{"project_name":"project1"}
```

标头和属性名称的区分大小写

借助 Amazon S3 Select，您可以使用双引号指示区分大小写的列标头（对于 CSV 对象）和属性（对于 JSON 对象）。如果没有双引号，则对象标头和属性不区分大小写。如果出现歧义，则会抛出错误。

以下示例是：1) 带有指定列标头并对于查询请求将 `FileHeaderInfo` 设置为 `"Use"` 的 CSV 格式的 Amazon S3 对象；或 2) 带有指定属性的 JSON 格式的 Amazon S3 对象。

示例 1：所查询的对象具有标头或属性 `NAME`。

- 以下表达式从对象成功返回值。因为没有引号，所以查询不区分大小写。

```
SELECT s.name from S3Object s
```

- 以下表达式导致出现 400 错误 `MissingHeaderName`。因为有引号，所以查询区分大小写。

```
SELECT s."name" from S3Object s
```

示例 2：所查询的 Amazon S3 对象具有一个包含 `NAME` 的标头或属性和另一个包含 `name` 的标头或属性。

- 以下表达式导致出现 400 错误 `AmbiguousFieldName`。由于无引号，所以查询不区分大小写，但有两个匹配项，因此引发错误。

```
SELECT s.name from S3Object s
```

- 以下表达式从对象成功返回值。因为有引号，所以查询区分大小写，因此没有歧义。

```
SELECT s."NAME" from S3Object s
```

将保留关键字用作用户定义的术语

Amazon S3 Select 具有一组保留关键字，在运行用于查询对象内容的 SQL 表达式时需要使用这组关键字。保留关键字包含函数名称、数据类型、运算符等。在某些情况下，诸如列标头（用于 CSV 文件）或属性（用于 JSON 对象）之类的用户定义的术语可能与保留关键字发生冲突。发生这种情况时，必须使用双引号指示您特意使用与保留关键字冲突的用户定义的术语。否则将会导致出现 400 解析错误。

有关保留关键字的完整列表，请参阅保留关键字。

以下示例是：1) 带有指定列标头和对于查询请求将 FileHeaderInfo 设置为 "Use" 的 CSV 格式的 Amazon S3 对象；或 2) 带有指定属性的 JSON 格式的 Amazon S3 对象。

示例：所查询的对象具有名为 CAST 的标头和属性，但这是保留关键字。

- 以下表达式从对象成功返回值。由于查询中使用了引号，因此 S3 Select 使用用户定义的标题或属性。

```
SELECT s."CAST" from S3Object s
```

- 以下表达式导致出现 400 解析错误。由于查询中不使用引号，因此 CAST 与保留关键字发生冲突。

```
SELECT s.CAST from S3Object s
```

标量表达式

在 WHERE 子句和 SELECT 列表中，您可以使用 SQL 标量表达式 (返回标量值的表达式)。它们具有以下形式：

- ***literal***

  SQL 文本。

- ***column_reference***

  对采用 *column_name* 或 *alias*.*column_name* 形式的列的引用。

- ***unary_op expression***

  在这种情况下，***unary_op*** 是一个 SQL 一元运算符。

- ***expression binary_op expression***

  在这种情况下，***binary_op*** 是一个 SQL 二进制运算符。

- ***func_name***

  在这种情况下，***func_name*** 是要调用的标量函数的名称。

- ***expression*** [NOT] BETWEEN ***expression*** AND ***expression***
- ***expression*** LIKE ***expression*** [ESCAPE ***expression***]

数据类型

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

Amazon S3 Select 支持多个基元数据类型。

数据类型转换

一般规则是遵循 CAST 函数（如果已定义）。如果未定义 CAST，则将所有输入数据视为字符串。在这种情况下，必要时，您必须将输入数据强制转换为相关的数据类型。

有关 CAST 函数的更多信息，请参阅CAST。

支持的数据类型

Amazon S3 Select 支持以下这组基元数据类型。

名称	说明	示例
bool	一个布尔值，可以是 TRUE 或 FALSE。	FALSE
int, integer	范围为 -9,223,372,036,854,775,808 到 9,223,372,036,854,775,807 的 8 字节有符号整数。	100000
string	UTF8 编码的长度可变的字符串。原定设置限制为 1 个字符。最大字符限制为 2,147,483,647。	'xyz'
float	8 字节浮点数。	CAST(0.456 AS FLOAT)
decimal, numeric	Base-10 数字，最大精度为 38（即最大有效位数），取值范围在 -2^{31} 到 $2^{31}-1$ 之间（即以 10 为底的指数）。	123.456

名称	说明	示例
	Note 如果您同时提供数值范围和精度，则 Amazon S3 Select 会全部忽略。	
`timestamp`	时间戳表示特定的时刻，始终包括本地偏移，并且能够支持任意精度。 在文本格式中，时间戳遵循关于日期和时间格式的 W3C 注释，但它们必须以文本 T 结尾（如果时间戳不是最低限度全天精度）。允许使用小数秒，具有至少一位精度，以及无限的最大值。本地时间偏移可以表示为与 UTC 相比的小时:分钟偏移量，或表示为文本 Z 以指示 UTC 的本地时间。本地时间偏移对于具有时间的时间戳是必需的，但对于日期值是不允许的。	`CAST('2007-04-05T14:30Z' AS TIMESTAMP)`

支持的 Parquet 类型

Amazon S3 Select 支持以下 Parquet 类型。

- `DATE`
- `DECIMAL`
- `ENUM`
- `INT(8)`
- `INT(16)`
- `INT(32)`
- `INT(64)`
- `LIST`

Note

对于 `LIST` Parquet 类型输出，Amazon S3 Select 仅支持 JSON 格式。但是，如果查询将数据限定为简单值，还可以用 CSV 格式查询 `LIST` Parquet 类型。

- STRING
- TIMESTAMP 支持的精度（MILLIS/MICROS/NANOS）

> **Note**
>
> 不支持保存为 INT(96) 的时间戳。
> 由于 INT(64) 类型的范围，使用 NANOS 单位的时间戳只能表示介于 1677-09-21 00:12:43 和 2262-04-11 23:47:16 之间的值。超出此范围的值不能用 NANOS 单位表示。

将 Parquet 类型映射到 Amazon S3 Select 中支持的数据类型

Parquet 类型	支持的数据类型
DATE	timestamp
DECIMAL	decimal, numeric
ENUM	string
INT(8)	int, integer
INT(16)	int, integer
INT(32)	int, integer
INT(64)	decimal, numeric
LIST	列表中的每个 Parquet 类型都映射到相应的数据类型。
STRING	string

Parquet 类型	支持的数据类型
TIMESTAMP	timestamp

运算符

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

Amazon S3 Select 支持以下运算符。

逻辑运算符

- AND
- NOT
- OR

比较运算符

- <
- >
- <=
- >=
- =
- <>
- !=
- BETWEEN
- IN - 例如：IN ('a', 'b', 'c')

模式匹配运算符

- LIKE

- _（匹配任何字符）
- %（匹配任何字符序列）

一元运算符

- IS NULL
- IS NOT NULL

数学运算符

支持加法、减法、乘法、除法和取模，如下所示：

- +
- -
- *
- /
- %

运算符优先顺序

下表按降序显示运算符的优先顺序。

运算符或元素	关联性	必需
-	右	一元减法
*, /, %	左	乘法、除法和取模
+, -	左	加法、减法
IN		设置成员资格
BETWEEN		范围包含
LIKE		字符串模式匹配

运算符或元素	关联性	必需
<>		小于、大于
=	右	等于、分配
NOT	右	逻辑非
AND	左	逻辑和
OR	左	逻辑或

保留关键字

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。[了解更多]

以下是适用于 Amazon S3 Select 的保留关键字列表。这些关键字包括函数名称、数据类型、运算符等，在运行用于查询对象内容的 SQL 表达式时需要使用它们。

```
absolute
action
add
all
allocate
alter
and
any
are
as
asc
assertion
at
authorization
avg
bag
begin
```

```
between
bit
bit_length
blob
bool
boolean
both
by
cascade
cascaded
case
cast
catalog
char
char_length
character
character_length
check
clob
close
coalesce
collate
collation
column
commit
connect
connection
constraint
constraints
continue
convert
corresponding
count
create
cross
current
current_date
current_time
current_timestamp
current_user
cursor
date
day
deallocate
```

```
dec
decimal
declare
default
deferrable
deferred
delete
desc
describe
descriptor
diagnostics
disconnect
distinct
domain
double
drop
else
end
end-exec
escape
except
exception
exec
execute
exists
external
extract
false
fetch
first
float
for
foreign
found
from
full
get
global
go
goto
grant
group
having
hour
```

```
identity
immediate
in
indicator
initially
inner
input
insensitive
insert
int
integer
intersect
interval
into
is
isolation
join
key
language
last
leading
left
level
like
limit
list
local
lower
match
max
min
minute
missing
module
month
names
national
natural
nchar
next
no
not
null
nullif
```

```
numeric
octet_length
of
on
only
open
option
or
order
outer
output
overlaps
pad
partial
pivot
position
precision
prepare
preserve
primary
prior
privileges
procedure
public
read
real
references
relative
restrict
revoke
right
rollback
rows
schema
scroll
second
section
select
session
session_user
set
sexp
size
smallint
```

```
some
space
sql
sqlcode
sqlerror
sqlstate
string
struct
substring
sum
symbol
system_user
table
temporary
then
time
timestamp
timezone_hour
timezone_minute
to
trailing
transaction
translate
translation
trim
true
tuple
union
unique
unknown
unpivot
update
upper
usage
user
using
value
values
varchar
varying
view
when
whenever
where
```

```
with
work
write
year
zone
```

SQL 函数

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

Amazon S3 Select 支持以下 SQL 函数。

主题

- 聚合函数
- 条件函数
- 转换函数
- 日期函数
- 字符串函数

聚合函数

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

Amazon S3 Select 支持以下聚合函数。

函数	参数类型	返回类型
AVG(*expressic* *n*)	INT, FLOAT, DECIMAL	DECIMAL 适用于 INT 参

函数	参数类型	返回类型
		数，FLOAT 适用于浮点型参数；否则与参数数据类型相同。
COUNT	-	INT
MAX(*expression*)	INT, DECIMAL	和参数类型相同。
MIN(*expression*)	INT, DECIMAL	和参数类型相同。
SUM(*expression*)	INT, FLOAT, DOUBLE, DECIMAL	INT 适用于 INT 参数，FLOAT 适用于浮点型参数；否则与参数数据类型相同。

SUM 示例

要聚合 S3 清单报告中某个文件夹的对象总大小，请使用 SUM 表达式。

以下 S3 清单报告是使用 GZIP 压缩的 CSV 文件。共有三列。

- 第一列是 S3 清单报告所用于的 S3 桶（*DOC-EXAMPLE-BUCKET*）的名称。
- 第二列是唯一标识桶中对象的对象键名称。

  第一行中的 *example-folder/* 值表示文件夹 *example-folder*。在 Amazon S3 中，当您在桶中创建文件夹时，S3 使用设置为您提供的文件夹名称的键创建一个 0 字节对象。

  第二行中的 *example-folder/object1* 值表示文件夹 *example-folder* 中的对象 *object1*。

  第三行中的 *example-folder/object2* 值表示文件夹 *example-folder* 中的对象 *object2*。

有关 S3 文件夹的更多信息，请参阅使用文件夹在 Amazon S3 控制台中整理对象。

- 第三列是对象大小（以字节为单位）。

```
"DOC-EXAMPLE-BUCKET","example-folder/","0"
"DOC-EXAMPLE-BUCKET","example-folder/object1","2011267"
"DOC-EXAMPLE-BUCKET","example-folder/object2","1570024"
```

要使用 SUM 表达式计算文件夹 *example-folder* 的总大小，请使用 Amazon S3 Select 运行 SQL 查询。

```
SELECT SUM(CAST(_3 as INT)) FROM s3object s WHERE _2 LIKE 'example-folder/%' AND _2 !=
 'example-folder/';
```

查询结果：

```
3581291
```

条件函数

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

Amazon S3 Select 支持以下条件函数。

主题

- CASE
- COALESCE
- NULLIF

CASE

CASE 表达式是一种条件表达式，类似于其他语言中的 if/then/else 语句。CASE 用于指定存在多个条件时的结果。有两种类型的 CASE 表达式：简单和搜索。

在简单 CASE 表达式中，将一个表达式与一个值比较。在找到匹配项时，将应用 THEN 子句中的指定操作。如果未找到匹配项，则应用 ELSE 子句中的操作。

在搜索 CASE 表达式中，基于布尔表达式计算每个 CASE，而且 CASE 语句会返回第一个匹配的 CASE。如果在 WHEN 子句中未找到匹配的 CASE，则返回 ELSE 子句中的操作。

语法

> **Note**
>
> 目前，Amazon S3 Select 不支持 ORDER BY 或包含新行的查询。请确保使用不带换行符的查询。

以下是用于匹配条件的简单 CASE 语句：

```
CASE expression WHEN value THEN result [WHEN...] [ELSE result] END
```

以下是用于计算每个条件的搜索 CASE 语句：

```
CASE WHEN boolean condition THEN result [WHEN ...] [ELSE result] END
```

示例

> **Note**
>
> 如果您使用 Amazon S3 控制台运行以下示例，并且 CSV 文件包含标题行，请选择排除第一行 CSV 数据。

示例 1：使用简单 CASE 表达式以在查询中将 New York City 替换为 Big Apple。将所有其他城市名称替换为 other。

```
SELECT venuecity, CASE venuecity WHEN 'New York City' THEN 'Big Apple' ELSE 'other' END
 FROM S3Object;
```

查询结果：

```
venuecity       |   case
```

```
-----------------+-----------
Los Angeles      | other
New York City    | Big Apple
San Francisco    | other
Baltimore        | other
...
```

示例 2：使用搜索 CASE 表达式来基于单个门票销售的 pricepaid 值分配组编号：

```
SELECT pricepaid, CASE WHEN CAST(pricepaid as FLOAT) < 10000 THEN 'group 1' WHEN
 CAST(pricepaid as FLOAT) > 10000 THEN 'group 2' ELSE 'group 3' END FROM S3Object;
```

查询结果：

```
pricepaid |  case
-----------+---------
12624.00 | group 2
10000.00 | group 3
10000.00 | group 3
9996.00 | group 1
9988.00 | group 1
...
```

COALESCE

COALESCE 按顺序评估参数并返回第一个非未知值，即第一个非空或非缺失值。此函数无法传播空值和缺失值。

语法

```
COALESCE ( expression, expression, ... )
```

参数

expression

对其执行函数的目标表达式。

示例

```
COALESCE(1)               -- 1
```

```
COALESCE(null)              -- null
COALESCE(null, null)        -- null
COALESCE(missing)           -- null
COALESCE(missing, missing) -- null
COALESCE(1, null)           -- 1
COALESCE(null, null, 1)     -- 1
COALESCE(null, 'string')    -- 'string'
COALESCE(missing, 1)        -- 1
```

NULLIF

给定两个表达式，如果两个表达式的计算结果为相同值，则 NULLIF 返回 NULL；否则 NULLIF 返回第一个表达式的计算结果。

语法

```
NULLIF ( expression1, expression2 )
```

参数

expression1, *expression2*

对其执行函数的目标表达式。

示例

```
NULLIF(1, 1)              -- null
NULLIF(1, 2)              -- 1
NULLIF(1.0, 1)            -- null
NULLIF(1, '1')            -- 1
NULLIF([1], [1])          -- null
NULLIF(1, NULL)           -- 1
NULLIF(NULL, 1)           -- null
NULLIF(null, null)        -- null
NULLIF(missing, null)     -- null
NULLIF(missing, missing) -- null
```

转换函数

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

Amazon S3 Select 支持以下转换函数。

主题

- CAST

CAST

CAST 函数将一种类型的实体 (如计算结果为单个值的表达式) 转换为另一种类型。

语法

```
CAST ( expression AS data_type )
```

参数

expression

一个或多个值、运算符和计算结果为值的 SQL 函数的组合。

data_type

要将表达式转换到的目标数据类型，如 INT。有关支持的数据类型的列表，请参阅 数据类型。

示例

```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)
CAST(0.456 AS FLOAT)
```

日期函数

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

Amazon S3 Select 支持以下日期函数。

主题

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

DATE_ADD

给定日期部分、数量和时间戳，DATE_ADD 返回通过修改由数量的日期部分而更新的时间戳。

语法

```
DATE_ADD( date_part, quantity, timestamp )
```

参数

date_part

指定要修改的日期部分。这可能是以下值之一：

- 年
- 个月
- day
- 小时
- minute

- 秒

quantity

应用到已更新的时间戳的值。*quantity* 的正值添加到时间戳的 date_part，并减去负值。

timestamp

对其执行函数的目标时间戳。

示例

```
DATE_ADD(year, 5, `2010-01-01T`)                -- 2015-01-01 (equivalent to
 2015-01-01T)
DATE_ADD(month, 1, `2010T`)                     -- 2010-02T (result will add precision
 as necessary)
DATE_ADD(month, 13, `2010T`)                    -- 2011-02T
DATE_ADD(day, -1, `2017-01-10T`)                -- 2017-01-09 (equivalent to
 2017-01-09T)
DATE_ADD(hour, 1, `2017T`)                      -- 2017-01-01T01:00-00:00
DATE_ADD(hour, 1, `2017-01-02T03:04Z`)          -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z
```

DATE_DIFF

给定日期部分和两个有效的时间戳，DATE_DIFF 返回日期部分的不同之处。当 *date_part* 的 *timestamp1* 值大于 *date_part* 的 *timestamp2* 值时，返回值为负整数。当 *date_part* 的 *timestamp1* 值小于 *date_part* 的 *timestamp2* 值时，返回值为正整数。

语法

```
DATE_DIFF( date_part, timestamp1, timestamp2 )
```

参数

date_part

指定要比较的时间戳部分。有关 date_part 的定义，请参阅DATE_ADD。

timestamp1

要比较的第一个时间戳。

timestamp2

要比较的第二个时间戳。

示例

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)            -- 1
DATE_DIFF(year, `2010T`, `2010-05T`)                     -- 4 (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
DATE_DIFF(month, `2010T`, `2011T`)                       -- 12
DATE_DIFF(month, `2011T`, `2010T`)                       -- -12
DATE_DIFF(day, `2010-01-01T23:00`, `2010-01-02T01:00`) -- 0 (need to be at least 24h
 apart to be 1 day apart)
```

EXTRACT

给定日期部分和时间戳，EXTRACT 返回时间戳的日期部分的值。

语法

```
EXTRACT( date_part FROM timestamp )
```

参数

date_part

指定时间戳中要提取的部分。这可能是以下值之一：

- YEAR
- MONTH
- DAY
- HOUR
- MINUTE
- SECOND
- TIMEZONE_HOUR
- TIMEZONE_MINUTE

timestamp

对其执行函数的目标时间戳。

示例

```
EXTRACT(YEAR FROM `2010-01-01T`)                                -- 2010
EXTRACT(MONTH FROM `2010T`)                                     -- 1 (equivalent to
 2010-01-01T00:00:00.000Z)
EXTRACT(MONTH FROM `2010-10T`)                                  -- 10
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`)                  -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`)                -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`)         -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`)       -- 8
```

TO_STRING

给定时间戳和格式模式，TO_STRING 以指定格式返回时间戳的字符串表示形式。

语法

```
TO_STRING ( timestamp time_format_pattern )
```

参数

timestamp

对其执行函数的目标时间戳。

time_format_pattern

具有以下特殊字符解释的字符串：

格式	示例	说明
yy	69	2 位年份
y	1969	4 位年份
yyyy	1969	以零填充的 4 位年份
M	1	一年中的月
MM	01	以零填充的一年中的月份

格式	示例	说明
MMM	Jan	一年中的月的缩写名称
MMMM	January	一年中的月的完整名称
MMMMM	J	一年中的月的首字母（注释：此格式与 TO_TIMESTAMP 函数结合使用时无效）
d	2	一个月中的日 (1-31)
dd	02	以零填充的一个月中的日 (01-31)
a	AM	一天的 AM 或 PM
h	3	一天中的小时 (1-12)
hh	03	以零填充的一天中的小时 (01-12)
H	3	一天中的小时 (0-23)
HH	03	以零填充的一天中的小时 (00-23)

格式	示例	说明
m	4	一小时中的分钟 (0-59)
mm	04	以零填充的一小时中的分钟 (00-59)
s	5	一分钟中的秒 (0-59)
ss	05	以零填充的一分钟中的秒 (00-59)
S	0	一秒钟的若干分之几（精度：0.1，范围：0.0-0.9）
SS	6	一秒钟的若干分之几（精度：0.01，范围：0.0-0.99）
SSS	60	一秒钟的若干分之几（精度：0.001，范围：0.0-0.999）
…	…	…

格式	示例	说明
SSSSSSSSS	60000000	一秒钟的若干分之几（最大精度：1 纳秒，范围：0.0-0.999999999）
n	60000000	纳秒级
X	+07 或 Z	小时数偏移，如果偏移为 0，则为 Z
XX 或 XXXX	+0700 或 Z	小时和分钟数偏移，如果偏移为 0，则为 Z
XXX 或 XXXXX	+07:00 或 Z	小时和分钟数偏移，如果偏移为 0，则为 Z
x	7	小时数偏移
xx 或 xxxx	700	小时和分钟数偏移
xxx 或 xxxxx	+07:00	小时和分钟数偏移

示例

```
TO_STRING(`1969-07-20T20:18Z`,  'MMMM d, y')                -- "July 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMM d, yyyy')               -- "Jul 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'M-d-yy')                    -- "7-20-69"
TO_STRING(`1969-07-20T20:18Z`, 'MM-d-y')                    -- "07-20-1969"
```

```
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y h:m a')              -- "July 20, 1969 8:18
 PM"
TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX')          --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX')    --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX')  --
 "1969-07-20T20:18:00+0800"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXXX') --
 "1969-07-20T20:18:00+08:00"
```

TO_TIMESTAMP

给定字符串，TO_TIMESTAMP 将其转换为时间戳。TO_TIMESTAMP 是 TO_STRING 的逆运算。

语法

```
TO_TIMESTAMP ( string )
```

参数

string

对其执行函数的目标字符串。

示例

```
TO_TIMESTAMP('2007T')                          -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

UTCNOW

UTCNOW 以时间戳的形式返回当前时间（UTC）。

语法

```
UTCNOW()
```

参数

UTCNOW 不带任何参数。

示例

```
UTCNOW() -- 2017-10-13T16:02:11.123Z
```

字符串函数

> **Important**
>
> 不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多

Amazon S3 Select 支持以下字符串函数。

主题

- CHAR_LENGTH, CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM
- UPPER

CHAR_LENGTH, CHARACTER_LENGTH

CHAR_LENGTH (或 CHARACTER_LENGTH) 计算指定字符串中的字符数。

> **Note**
>
> CHAR_LENGTH 和 CHARACTER_LENGTH 是同义词。

语法

```
CHAR_LENGTH ( string )
```

参数

string

对其执行函数的目标字符串。

示例

```
CHAR_LENGTH('')          -- 0
CHAR_LENGTH('abcdefg')   -- 7
```

LOWER

给定字符串，LOWER 将所有大写字符转换为小写字符。所有非大写字符保持不变。

语法

```
LOWER ( string )
```

参数

string

对其执行函数的目标字符串。

示例

```
LOWER('AbCdEfG!@#$') -- 'abcdefg!@#$'
```

SUBSTRING

指定字符串、起始索引和长度（可选），SUBSTRING 返回从起始索引直至字符串结尾处的子字符串，或最大长度为输入的字符串长度的子字符串。

> **Note**
>
> 输入字符串的首字符的索引位置为 1。
>
> - 如果 start < 1，且未指定长度，则索引位置设置为 1。

- 如果 start < 1，且指定了长度，则索引位置设置为 start + length -1。
- 如果 start + length -1 < 0，则返回一个空字符串。
- 如果 start + length -1 > = 0，则返回从索引位置 1 开始的长度为 start + length - 1 的子字符串。

语法

```
SUBSTRING( string FROM start [ FOR length ] )
```

参数

string

对其执行函数的目标字符串。

start

字符串的开始位置。

length

要返回的子字符串的长度。如果不存在，则执行到字符串的结尾。

示例

```
SUBSTRING("123456789", 0)      -- "123456789"
SUBSTRING("123456789", 1)      -- "123456789"
SUBSTRING("123456789", 2)      -- "23456789"
SUBSTRING("123456789", -4)     -- "123456789"
SUBSTRING("123456789", 0, 999) -- "123456789"
SUBSTRING("123456789", 1, 5)   -- "12345"
```

TRIM

从字符串中剪裁前导或尾随字符。要删除的原定设置字符为空格（' '）。

语法

```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

参数

string

对其执行函数的目标字符串。

LEADING | TRAILING | BOTH

此参数指示是否剪裁前导和/或尾随字符。

remove_chars

要删除的一组字符。*remove_chars* 可能是长度 > 1 的字符串。此函数返回包含在已删除字符串开头或结尾发现的 *remove_chars* 中任何字符的字符串。

示例

```
TRIM('       foobar         ')               -- 'foobar'
TRIM('      \tfoobar\t         ')            -- '\tfoobar\t'
TRIM(LEADING FROM '       foobar         ')  -- 'foobar         '
TRIM(TRAILING FROM '       foobar         ') -- '       foobar'
TRIM(BOTH FROM '       foobar         ')     -- 'foobar'
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'
```

UPPER

给定字符串，UPPER 将所有小写字符转换为大写字符。所有非小写字符保持不变。

语法

```
UPPER ( string )
```

参数

string

对其执行函数的目标字符串。

示例

```
UPPER('AbCdEfG!@#$') -- 'ABCDEFG!@#$'
```

对 Amazon S3 对象执行大规模批量操作

您可以使用 S3 批量操作对 Amazon S3 对象执行大规模批量操作。S3 批量操作可以对您指定的 Amazon S3 对象列表执行单个操作。单个任务可对数十亿个包含 EB 级数据的对象执行指定操作。Amazon S3 跟踪进度、发送通知并存储所有操作的详细完成报告，从而提供完全托管、可审核的无服务器体验。您可以通过 AWS Management Console、AWS CLI、Amazon SDK 或 REST API 使用 S3 批量操作。

使用 S3 批量操作复制对象并设置对象标签或访问控制列表 (ACL)。您还可以从 S3 Glacier Flexible Retrieval 启动对象还原，或者调用 AWS Lambda 函数以使用您的对象执行自定义操作。您可以对自定义的对象列表执行这些操作，也可以使用 Amazon S3 清单报告来轻松生成对象列表。Amazon S3 批量操作使用与 Amazon S3 相同的 Amazon S3 API，因此您会发现界面很熟悉。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和 目录桶。有关将批量操作与 S3 Express One Zone 和目录存储桶配合使用的更多信息，请参阅对 S3 Express One Zone 使用批量操作。

S3 批量操作基础知识

您可以使用 S3 批量操作对 Amazon S3 对象执行大规模批量操作。S3 批量操作可以对您指定的 Amazon S3 对象列表运行单个操作。

术语

此部分使用的术语任务、操作 和任务 定义如下：

任务

任务是 S3 批量操作的基本工作单位。任务包含对清单中所列对象运行指定操作所需的全部信息。在您提供此信息并请求任务开始后，任务将对清单中的每个对象执行操作。

操作

操作是您希望批量操作任务运行的 API 操作的类型，例如复制对象。对于清单中指定的所有对象，每个任务执行单一类型的操作。

任务

任务是任务的执行单元。任务表示为了对单个对象执行任务的操作而对 Amazon S3 或 AWS Lambda API 操作进行的单个调用。在任务的生命周期内，S3 批量操作将为清单中指定的每个对象创建一个任务。

S3 批量操作任务的工作原理

任务是 S3 批量操作的基本工作单位。任务包含要对对象列表运行指定操作所需的全部信息。要创建任务，您需要向 S3 批量操作提供对象列表并指定要对这些对象执行的操作。

有关 S3 批量操作支持的操作的信息，请参阅 S3 分批操作支持的操作。

批处理任务将对其清单中包含的每个对象执行指定操作。清单列出了您希望批处理任务进行处理的对象并将其作为对象存储在存储桶中。您可以使用逗号分隔值 (CSV) 格式的 Amazon S3 清单 报告作为清单，这样便于轻松创建位于存储桶中的大型对象列表。您还可以指定采用简单的 CSV 格式的清单，这样您便可对单个存储桶中包含的对象的自定义列表执行批量操作。

在您创建任务后，Amazon S3 将处理清单中的对象列表并对每个对象运行指定的操作。在任务运行时，您能够通过编程方式或通过 Amazon S3 控制台监控其进度。您还可以对任务进行配置，以便在其完成时生成完成报告。完成报告会描述任务已执行的每个任务的结果。有关监控任务的更多信息，请参阅 管理 S3 分批操作任务。

S3 批量操作教程

以下教程提供了一些批量操作任务的完整端到端过程。

- 教程：使用 S3 批量操作、AWS Lambda 和 AWS Elemental MediaConvert 对视频进行批量转码

授予 Amazon S3 分批操作的权限

在创建和运行 S3 分批操作任务之前，您必须授予所需的权限。要创建 Amazon S3 分批操作任务，需要 `s3:CreateJob` 用户权限。创建任务的同一个实体也必须具有 `iam:PassRole` 权限，以便将为此任务指定的 AWS Identity and Access Management (IAM) 角色传递到分批操作。

有关指定 IAM 资源的一般信息，请参阅 IAM 用户指南中的 IAM JSON 策略 – 资源元素。以下各节提供了有关创建 IAM 角色和附加策略的信息。

主题

- 创建 S3 分批操作 IAM 角色

- 附加权限策略

创建 S3 分批操作 IAM 角色

Amazon S3 必须具有权限才能代表您执行 S3 分批操作。您通过 AWS Identity and Access Management (IAM) 角色授予这些权限。此部分提供您在创建 IAM 角色时使用的信任和权限策略的示例。有关更多信息，请参阅 IAM 用户指南中的 IAM 角色。有关示例，请参阅 使用任务标签控制 S3 分批操作的权限 和 使用 S3 分批操作复制对象。

在 IAM 策略中，您还可以使用条件键筛选 S3 分批操作任务的访问权限。有关更多信息和 Amazon S3 特定条件键的完整列表，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

信任策略

要允许 S3 分批操作服务委托人担任 IAM 角色，您可将以下信任策略附加到该角色。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

附加权限策略

根据操作的类型，您可以附加以下策略之一。

在配置权限之前，请注意以下事项：

- 不论执行什么操作，Amazon S3 都需要权限来从您的 S3 存储桶读取清单对象并（可选）将报告写入您的存储桶。因此，所有以下策略均包含这些权限。
- 对于 Amazon S3 清单报告清单，S3 分批操作需要读取 manifest.json 对象以及所有关联的 CSV 数据文件的权限。
- 当您指定对象的版本 ID 时，只需要版本特定的权限，如 `s3:GetObjectVersion`。

- 如果您在加密对象上运行 S3 分批操作，则 IAM 角色还必须拥有对用于加密这些对象的 AWS KMS 密钥的访问权限。
- 如果您提交使用 AWS KMS 加密的清单报告清单，则 IAM 策略必须包含对于 manifest.json 对象以及所有关联 CSV 数据文件的 `"kms:Decrypt"` 和 `"kms:GenerateDataKey"` 权限。
- 如果批量操作任务在已启用 ACL 且位于其它 AWS 账户内的存储桶中生成清单，您必须在为该批量任务配置的 IAM 角色的 IAM 策略中授予 `s3:PutObjectAcl` 权限。如果您没有包含此权限，则批量任务会失败，并显示错误 `Error occurred when preparing manifest: Failed to write manifest`。

复制对象：PutObject

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::DestinationBucket/*"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::SourceBucket",
                "arn:aws:s3:::SourceBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

替换对象标签：PutObjectTagging

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
```

```
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

删除对象标签：DeleteObjectTagging

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "s3:DeleteObjectTagging",
              "s3:DeleteObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
```

```
    ]
}
```

替换访问控制列表：PutObjectAcl

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

还原对象：RestoreObject

```
{
  "Version":"2012-10-17",
  "Statement":[
```

```
    {
      "Effect":"Allow",
      "Action":[
          "s3:RestoreObject"
      ],
      "Resource": "arn:aws:s3:::TargetResource/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::ManifestBucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::ReportBucket/*"
      ]
    }
  ]
}
```

应用对象锁定保留：PutObjectRetention

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::TargetResource"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:PutObjectRetention",
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

应用对象锁定依法保留：PutObjectLegalHold

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::TargetResource"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": "s3:PutObjectLegalHold",
            "Resource": [
                "arn:aws:s3:::TargetResource/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ReportBucket/*"
            ]
        }
    ]
}
```

复制现有对象：使用 S3 生成的清单启动复制

如果使用和存储 S3 生成的清单，请使用此策略。有关使用分批操作复制现有对象的更多信息，请参阅使用 S3 分批复制以复制现有对象。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
```

```
         ]
      },
      {
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*",
            "arn:aws:s3:::*** manifest bucket ****/*"
         ]
      }
   ]
}
```

复制现有对象：使用用户清单启动复制

使用用户提供的清单时使用此策略。有关使用分批操作复制现有对象的更多信息，请参阅 使用 S3 分批复制以复制现有对象。

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** replication source bucket ***/*"
         ]
      },
            {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::*** manifest bucket ***/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::*** completion report bucket ****/*"
         ]
      }
   ]
}
```

创建 S3 批量操作任务

借助 Amazon S3 批量操作，您可以对特定 Amazon S3 对象的列表执行大规模批量操作。本节介绍创建 S3 批量操作任务所需的信息以及 `CreateJob` 请求的结果。它还提供说明，讲解如何通过使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK for Java 创建批量操作任务。

当您创建 S3 批量操作任务时，可以为全部任务或仅失败任务请求完成报告。只要成功调用了至少一个任务，S3 批量操作即会为已完成、已失败或已取消的任务生成报告。有关更多信息，请参阅 示例：S3 分批操作完成报告。

主题

批量操作任务请求元素

要创建 S3 批量操作任务，您必须提供以下信息：

操作

指定希望 S3 批量操作对清单中的对象运行的操作。每种操作类型都接受特定于该操作的参数。借助批量操作功能，您可以批量执行操作，其结果与对每个对象逐个执行该操作相同。

清单

清单是您希望 S3 批量操作对其运行指定操作的所有对象的列表。您可以使用以下方法为批量操作任务指定清单。

- 手动创建您自己自定义的 CSV 格式的对象列表。
- 选择现有的 CSV 格式的 Amazon S3 清单 报告。
- 直接批量操作功能可根据您在创建任务时指定的对象筛选条件自动生成清单。此选项适用于您在 Amazon S3 控制台中创建的批量复制任务，也适用于您使用 AWS CLI、AWS SDK 或 Amazon S3 REST API 创建的任何任务类型。

> **Note**
>
> - 无论您如何指定清单，列表本身都必须存储在通用存储桶中。批量操作无法从目录存储桶中导入现有清单，也无法将生成的清单保存到目录存储桶。但是，清单中描述的对象可以存储在目录存储桶中。有关更多信息，请参阅目录存储桶。
> - 如果清单中的对象位于受版本控制的存储桶中，指定对象的版本 ID 时，批量操作将在特定版本上执行此操作。如果未指定版本 ID，则批量操作将针对最新对象版本执行该操作。如果您的清单包含版本 ID 字段，则必须为清单中的所有对象提供版本 ID。

有关更多信息，请参阅 指定清单。

优先级

使用任务优先级指示此任务相对于在您账户中运行的其他任务的优先级。数字越大，优先级越高。

任务优先级仅相对于为同一账户和区域中的其他任务设置的优先级有意义。您可以选择适合您的编号系统。例如，您可能希望将所有还原（`RestoreObject`）任务的优先级指定为 1，将所有复制（`CopyObject`）任务的优先级指定为 2，并将所有替换访问控制列表（ACL）（`PutObjectAcl`）任务的优先级指定为 3。

S3 批量操作根据优先级编号确定任务的优先级，但不严格保证顺序。因此，任务优先级并不用来确保任何一个任务会在其他任何任务之前开始或完成。如果您需要确保严格的顺序，请等到一个任务完成后，再开始下一个任务。

RoleArn

指定将运行任务的 AWS Identity and Access Management (IAM) 角色。您使用的 IAM 角色必须具有足够的权限来执行任务中指定的操作。例如，要运行 `CopyObject` 任务，IAM 角色必须具有针对源存储桶的 `s3:GetObject` 权限和针对目标存储桶的 `s3:PutObject` 权限。该角色还需要读取清单和写入任务完成报告的权限。

有关 IAM 角色的更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

有关 Amazon S3 权限的更多信息，请参阅 Amazon S3 的策略操作。

> **Note**
>
> 对目录存储桶执行操作的批量操作任务需要特定的权限。有关更多信息，请参阅适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

报告

指定您是否希望 S3 批量操作生成完成报告。如果您请求任务完成报告，还必须在此元素中提供报告参数。必要的信息包括：

- 您希望存储报告的存储桶

  > **Note**
  >
  > 报告必须存储在通用存储桶中。批量操作功能无法将报告保存到目录存储桶中。有关更多信息，请参阅目录存储桶。

- 报告的格式
- 您希望报告包含所有任务的详细信息还是只包含失败的任务

- 可选的前缀字符串

> **Note**
>
> 完成报告始终使用 Amazon S3 托管式密钥（SSE-S3）进行加密。

标签（可选）

您可以通过添加标签来标记和控制对 S3 批量操作任务的访问。您可以使用标签来标识谁负责批量操作任务，或者控制用户与批量操作任务的交互方式。任务标签的存在可授予或限制用户取消任务、激活处于确认状态的任务或更改任务优先级的能力。例如，您可以授予用户调用 `CreateJob` 操作的权限，前提是使用标签 `"Department=Finance"` 创建此任务。

您可以创建附加了标签的任务，并且可以在创建任务后向任务添加标签。

有关更多信息，请参阅 the section called “使用标签”。

Description（可选）

要跟踪和监控任务，您还可以提供最多包含 256 个字符的描述。只要返回有关任务的信息或在 Amazon S3 控制台中显示任务详细信息，Amazon S3 便会包含此描述。随后您便可以根据自己分配的描述轻松地对任务进行排序和筛选。描述不一定是唯一的，因此您可以使用描述作为类别（例如“每周日志复制任务”）来帮助您跟踪多组相似的任务。

指定清单

清单是一种 Amazon S3 对象，其中包含您希望 Amazon S3 采取操作的对象键。您可以通过下列方式之一提供清单：

- 手动创建新的清单文件。
- 使用现有的清单。
- 直接批量操作功能可根据您在创建任务时指定的对象筛选条件自动生成清单。此选项适用于您在 Amazon S3 控制台中创建的批量复制任务，也适用于您使用 AWS CLI、AWS SDK 或 Amazon S3 REST API 创建的任何任务类型。

> **Note**
>
> 无论您如何指定清单，列表本身都必须存储在通用存储桶中。批量操作无法从目录存储桶中导入现有清单，也无法将生成的清单保存到目录存储桶。但是，清单中描述的对象可以存储在目录存储桶中。有关更多信息，请参阅目录存储桶。

创建清单文件

要手动创建清单文件，您需要采用 CSV 格式列表的形式，指定清单对象键、ETag（实体键）和可选的版本 ID。清单的内容必须是 URL 编码的。

默认情况下，Amazon S3 自动使用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）对上传到 Amazon S3 存储桶的清单进行加密。不支持使用具有客户提供密钥（SSE-C）的服务器端加密的清单。仅当使用 CSV 格式的清单报告时，才支持使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）的清单。不支持将手动创建的清单与 AWS KMS 一起使用。

您的清单必须包括存储桶名称、对象键以及（可选）每个对象的对象版本。S3 批量操作不使用清单中的任何其他字段。

> **Note**
>
> 如果清单中的对象位于受版本控制的存储桶中，指定对象的版本 ID 时，批量操作将在特定版本上执行此操作。如果未指定版本 ID，则批量操作将针对最新对象版本执行该操作。如果您的清单包含版本 ID 字段，则必须为清单中的所有对象提供版本 ID。

下面是 CSV 格式的没有版本 ID 的清单示例。

```
Examplebucket,objectkey1
Examplebucket,objectkey2
Examplebucket,objectkey3
Examplebucket,photos/jpgs/objectkey4
Examplebucket,photos/jpgs/newjersey/objectkey5
Examplebucket,object%20key%20with%20spaces
```

下面是 CSV 格式的包含版本 ID 的清单示例。

```
Examplebucket,objectkey1,PZ9ibn9D5lP6p298B7S9_ceqx1n5EJ0p
Examplebucket,objectkey2,YY_ouuAJByNW1LRBfFMfxMge7XQWxMBF
```

```
Examplebucket,objectkey3,jbo9_jhdPEyB4RrmOxWS0kU0EoNrU_oI
Examplebucket,photos/jpgs/objectkey4,6EqlikJJxLTsHsnbZbSRffn24_eh5Ny4
Examplebucket,photos/jpgs/newjersey/objectkey5,imHf3FAiRsvBW_EHB8GOu.NHunHO1gVs
Examplebucket,object%20key%20with%20spaces,9HkPvDaZY5MVbMhn6TMn1YTb5ArQAo3w
```

指定现有的清单文件

您可以使用以下两种格式之一，在创建任务请求中指定清单文件：

- Amazon S3 清单报告 – 必须是 CSV 格式的 Amazon S3 清单报告。您必须指定与清单报告关联的 `manifest.json` 文件。有关清单报告的更多信息，请参阅 Amazon S3 清单。如果清单报告包括版本 ID，S3 批量操作将对特定对象版本进行操作。

  > **Note**
  >
  > - S3 批量操作支持用 SSE-KMS 加密的 CSV 格式清单报告。
  > - 如果您提交使用 SSE-KMS 加密的清单报告清单，则 IAM 策略必须包含对于 `manifest.json` 对象以及所有关联 CSV 数据文件的 `"kms:Decrypt"` 和 `"kms:GenerateDataKey"` 权限。

- CSV 文件 – 文件中的每一行必须包括存储桶名称和对象键，还可选择包括对象版本。对象键必须进行 URL 编码，如以下示例所示。清单必须包含所有对象的版本 ID 或忽略所有对象的版本 ID。有关 CSV 清单格式的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 JobManifestSpec。

  > **Note**
  >
  > S3 批量操作不支持使用 SSE-KMS 加密的 CSV 清单文件。

> **Important**
>
> 当您使用手动创建的清单和受版本控制的存储桶时，我们建议您指定对象的版本 ID。创建任务时，S3 批量操作会在运行任务之前解析整个清单。不过，它不会获取存储桶状态的“快照”。由于清单可以包含数十亿个对象，因此任务可能需要很长时间才能运行完，这可能会影响任务所针对的对象版本。假设您在任务运行时用新版本覆盖某个对象，但未指定该对象的版本 ID。在这种情况下，Amazon S3 将对该对象的最新版本（而不是在创建作业时存在的版本）执行操作。避免此行为的唯一方式是为清单中列出的对象指定版本 ID。

自动生成清单

您可以指示 Amazon S3 批量操作根据创建任务时指定的对象筛选条件自动生成清单。此选项适用于您在 Amazon S3 控制台中创建的批量复制任务，也适用于您使用 AWS CLI、AWS SDK 或 Amazon S3 REST API 创建的任何任务类型。有关分批复制的更多信息，请参阅 使用 S3 分批复制以复制现有对象。

要自动生成清单，请在任务创建请求中指定以下元素：

- 有关包含您的源对象的存储桶的信息，包括存储桶拥有者和 Amazon 资源名称（ARN）
- 有关清单输出的信息，包括用于创建清单文件的标志、输出存储桶拥有者、ARN、前缀、文件格式和加密类型
- 按创建日期、密钥名称、大小、存储类和标签来筛选对象的可选标准

对象筛选标准

要筛选要包含在自动生成的清单中的对象列表，您可以指定以下标准。有关更多信息，请参阅《Amazon S3 API 参考》中的 JobManifestGeneratorFilter。

CreatedAfter

如果提供，则生成的清单仅包含在此时间之后创建的源存储桶对象。

CreatedBefore

如果提供，则生成的清单仅包含在此时间之前创建的源存储桶对象。

EligibleForReplication

如果提供，则生成的清单仅包含根据源存储桶上的复制配置而符合复制条件的对象。

KeyNameConstraint

如果提供，则生成的清单仅包含下面这样的源存储桶对象：其对象密钥与 MatchAnySubstring、MatchAnyPrefix 和 MatchAnySuffix 所指定的字符串约束相匹配。

MatchAnySubstring – 如果提供，则当指定的字符串出现在对象密钥字符串中的任何位置时，生成的清单将包含对象。

MatchAnyPrefix – 如果提供，则当指定的字符串出现在对象密钥字符串的开头时，生成的清单将包含对象。

MatchAnySuffix – 如果提供，则当指定的字符串出现在对象密钥字符串的末尾时，生成的清单将包含对象。

MatchAnyStorageClass

如果提供，则生成的清单仅包含以指定存储类存储的源存储桶对象。

ObjectReplicationStatuses

如果提供，则生成的清单仅包含具有指定复制状态之一的源存储桶对象。

ObjectSizeGreaterThanBytes

如果提供，则生成的清单仅包含文件大小大于指定字节数的源存储桶对象。

ObjectSizeLessThanBytes

如果提供，则生成的清单仅包含文件大小小于指定字节数的源存储桶对象。

> **Note**
>
> 您无法克隆大多数自动生成清单的任务。可以克隆批量复制任务，除非它们使用 `KeyNameConstraint`、`MatchAnyStorageClass`、`ObjectSizeGreaterThanBytes` 或 `ObjectSizeLessThanBytes` 清单筛选标准。

指定清单标准的语法因您用于创建任务的方法而异。有关示例，请参阅创建作业。

创建作业

您可以使用 Amazon S3 控制台、AWS CLI、AWS SDK 或 Amazon S3 REST API 创建 S3 批量操作任务。

有关创建任务请求的更多信息，请参阅 批量操作任务请求元素。

先决条件

在创建批量操作任务之前，请确认您已配置相关权限。有关更多信息，请参阅 授予 Amazon S3 分批操作的权限。

使用 S3 控制台

创建批量任务

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要在其中创建任务的区域。

   > **Note**
   >
   > 对于复制操作，必须在目标存储桶所在的同一区域中创建任务。对于所有其它操作，必须在与清单中的对象相同的区域中创建任务。

3. 在 Amazon S3 控制台的左侧导航窗格中选择批量操作。
4. 请选择创建任务。
5. 查看要在其中创建任务的 AWS 区域。
6. 在 Manifest format (清单格式) 下，请选择要使用的清单对象的类型。

   - 如果您选择 S3 inventory report (S3 清单报告)，请输入 Amazon S3 在 CSV 格式的清单报告中生成的 manifest.json 对象的路径，以及（可选）输入清单对象的版本 ID（如果您要使用的版本不是最新版本）。
   - 如果您选择 CSV，请输入 CSV 格式清单对象的路径。清单对象必须遵循控制台中描述的格式。如果您希望使用并非最新的对象版本，则可以选择包含清单对象的版本 ID。

   > **Note**
   >
   > Amazon S3 控制台仅支持对批量复制任务自动生成清单。对于其他所有任务类型，如果您希望 Amazon S3 根据您指定的筛选标准自动生成清单，则必须使用 AWS CLI、AWS SDK 或 Amazon S3 REST API 配置您的任务。

7. 请选择 Next（下一步）。
8. 在 Operation (操作) 下，请选择您希望对清单中列出的所有对象执行的操作。填写您所选操作的信息，然后选择下一步。
9. 填写配置额外选项的信息，然后选择下一步。

10. 对于审核，验证设置。如果需要进行更改，请选择 Previous。否则，请选择 Create job (创建任务)。

使用 AWS CLI

Specify manifest

以下示例说明如何创建 S3 批量操作 `S3PutObjectTagging` 任务，以便处理现有清单文件中列出的对象。

创建批量操作 **`S3PutObjectTagging`** 任务

1. 使用以下命令创建 AWS Identity and Access Management（IAM）角色，然后创建一个 IAM 策略来分配相关权限。以下角色和策略授予 Amazon S3 添加对象标签的权限，您在后续步骤中创建任务时将需要用到该权限。

   a. 使用以下示例命令创建 IAM 角色供批量操作使用。要使用此示例命令，请将 *`S3BatchJobRole`* 替换为要为角色指定的名称。

```
aws iam create-role \
 --role-name S3BatchJobRole \
 --assume-role-policy-document '{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}'
```

   记下角色的 Amazon 资源名称（ARN）。创建任务时，您需要该 ARN。

   b. 使用以下示例命令创建具有必要权限的 IAM 策略，并将其附加到您在上一步中创建的 IAM 角色。有关必要权限的更多信息，请参阅[授予 Amazon S3 分批操作的权限]。

> **Note**
>
> 对目录存储桶执行操作的批量操作任务需要特定的权限。有关更多信息，请参阅适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

要使用此示例命令，请按如下方式替换 *user input placeholders*：

- 将 *S3BatchJobRole* 替换为您的 IAM 角色的名称。请确保此名称与您之前使用的名称一致。
- 将 *PutObjectTaggingBatchJobPolicy* 替换为要为 IAM 策略指定的名称。
- 将 *amzn-s3-demo-destination-bucket* 替换为包含要应用标签的对象的存储桶的名称。
- 将 *DOC-EXAMPLE-MANIFEST-BUCKET* 替换为包含清单的存储桶的名称。
- 将 *DOC-EXAMPLE-REPORT-BUCKET* 替换为要将完成报告交付到的存储桶的名称。

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name PutObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-BUCKET/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET/*"
      ]
    }
  ]
}'
```

2. 使用以下示例命令创建 S3PutObjectTagging 任务。

manifest.csv 文件提供存储桶和对象键值的列表。任务将指定的标签应用到清单中标识的对象。ETag 是 manifest.csv 对象的 ETag，可以从 Amazon S3 控制台获取。此请求指定了 no-confirmation-required 参数，这样您就可以运行任务，而不必使用 update-job-status 命令进行确认。有关更多信息，请参阅 AWS CLI 命令参考 中的 create-job。

要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。将 *IAM-role* 替换为您之前创建的 IAM 角色的 ARN。

```
aws s3control create-job \
    --region us-west-2 \
    --account-id acct-id \
    --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"keyOne",
 "Value":"ValueOne"}] }}' \
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":
{"ObjectArn":"arn:aws:s3:::my_manifests/
manifest.csv","ETag":"60e460c9d1046e73f7dde5043ac3ae85"}}' \
    --report '{"Bucket":"arn:aws:s3:::DOC-EXAMPLE-REPORT-
BUCKET","Prefix":"final-reports",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
    --priority 42 \
    --role-arn IAM-role \
    --client-request-token $(uuidgen) \
```

```
    --description "job description" \
    --no-confirmation-required
```

作为响应，Amazon S3 返回任务 ID（例如 `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`）。您需要使用任务 ID 来识别、监控和修改任务。

Generate manifest

以下示例说明如何创建 S3 批量操作 `S3DeleteObjectTagging` 任务，该任务会根据您的对象筛选标准自动生成清单。此标准包括创建日期、密钥名称、大小、存储类和标签。

创建批量操作 **`S3DeleteObjectTagging`** 任务

1. 使用以下命令创建 AWS Identity and Access Management（IAM）角色，然后创建一个 IAM 策略来分配权限。以下角色和策略授予 Amazon S3 删除对象标签的权限，您在后续步骤中创建任务时将需要用到该权限。

   a. 使用以下示例命令创建 IAM 角色供批量操作使用。要使用此示例命令，请将 *`S3BatchJobRole`* 替换为要为角色指定的名称。

   ```
   aws iam create-role \
    --role-name S3BatchJobRole \
    --assume-role-policy-document '{
      "Version":"2012-10-17",
      "Statement":[
         {
            "Effect":"Allow",
            "Principal":{
               "Service":"batchoperations.s3.amazonaws.com"
            },
            "Action":"sts:AssumeRole"
         }
      ]
   }'
   ```

   记下角色的 Amazon 资源名称（ARN）。创建任务时，您需要该 ARN。

   b. 使用以下示例命令创建具有必要权限的 IAM 策略，并将其附加到您在上一步中创建的 IAM 角色。有关必要权限的更多信息，请参阅[授予 Amazon S3 分批操作的权限]。

> **Note**
>
> 对目录存储桶执行操作的批量操作任务需要特定的权限。有关更多信息，请参阅适用于 S3 Express One Zone 的 AWS Identity and Access Management（IAM）。

要使用此示例命令，请按如下方式替换 *user input placeholders*：

- 将 *S3BatchJobRole* 替换为您的 IAM 角色的名称。请确保此名称与您之前使用的名称一致。
- 将 *DeleteObjectTaggingBatchJobPolicy* 替换为要为 IAM 策略指定的名称。
- 将 *amzn-s3-demo-destination-bucket* 替换为包含要应用标签的对象的存储桶的名称。
- 将 *DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET* 替换为您想用于保存清单的存储桶的名称。
- 将 *DOC-EXAMPLE-REPORT-BUCKET* 替换为要将完成报告交付到的存储桶的名称。

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name DeleteObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:DeleteObjectTagging",
        "s3:DeleteObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutInventoryConfiguration"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:ListBucket"
      ],
      "Resource":[
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET/*",
        "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET/*"
      ]
    }
  ]
}'
```

2. 使用以下命令创建 S3DeleteObjectTagging 任务。

   在此示例中，--report 部分中的值指定了将生成的任务报告的存储桶、前缀、格式和范围。该 --manifest -generator 部分指定了有关包含任务将处理的对象的源存储桶的信息、有关将为任务生成的清单输出列表的信息，以及按创建日期、名称限制、大小和存储类来缩小要包含在清单中的对象范围的筛选标准。该命令还指定任务的优先级、IAM 角色和 AWS 区域。

   有关更多信息，请参阅 AWS CLI 命令参考 中的 create-job。

   要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。将 *IAM-role* 替换为您之前创建的 IAM 角色的 ARN。

```
aws s3control create-job \
    --account-id 012345678901 \
    --operation '{
```

```
        "S3DeleteObjectTagging": {}
    }' \
    --report '{
        "Bucket":"arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET",
        "Prefix":"reports",
        "Format":"Report_CSV_20180820",
        "Enabled":true,
        "ReportScope":"AllTasks"
    }' \
    --manifest-generator '{
        "S3JobManifestGenerator": {
          "ExpectedBucketOwner": "012345678901",
          "SourceBucket": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
          "EnableManifestOutput": true,
          "ManifestOutputLocation": {
            "ExpectedManifestBucketOwner": "012345678901",
            "Bucket": "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-OUTPUT-BUCKET",
            "ManifestPrefix": "prefix",
            "ManifestFormat": "S3InventoryReport_CSV_20211130"
          },
          "Filter": {
            "CreatedAfter": "2023-09-01",
            "CreatedBefore": "2023-10-01",
            "KeyNameConstraint": {
              "MatchAnyPrefix": [
                "prefix"
              ],
              "MatchAnySuffix": [
                "suffix"
              ]
            },
            "ObjectSizeGreaterThanBytes": 100,
            "ObjectSizeLessThanBytes": 200,
            "MatchAnyStorageClass": [
              "STANDARD",
              "STANDARD_IA"
            ]
          }
        }
      }' \
     --priority 2 \
     --role-arn IAM-role \
     --region us-east-1
```

作为响应，Amazon S3 返回任务 ID（例如 `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`）。您将需要此任务 ID 来识别、监控或修改任务。

使用 AWS SDK for Java

Specify manifest

以下示例说明如何创建 S3 批量操作 `S3PutObjectTagging` 任务，以便处理现有清单文件中列出的对象。要使用此示例，请将 *`user input placeholders`* 替换为您自己的信息。

Example

```
package aws.example.s3control;



import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.*;

import java.util.UUID;
import java.util.ArrayList;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateJob {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String iamRoleArn = "IAM Role ARN";
        String reportBucketName = "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET";
        String uuid = UUID.randomUUID().toString();

        ArrayList tagSet = new ArrayList<S3Tag>();
        tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));


        try {
            JobOperation jobOperation = new JobOperation()
```

```
                .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                        .withTagSet(tagSet)
                );

        JobManifest manifest = new JobManifest()
                .withSpec(new JobManifestSpec()
                        .withFormat("S3BatchOperations_CSV_20180820")
                        .withFields(new String[]{
                                "Bucket", "Key"
                        }))
                .withLocation(new JobManifestLocation()
                        .withObjectArn("arn:aws:s3:::my_manifests/manifest.csv")
                        .withETag("60e460c9d1046e73f7dde5043ac3ae85"));
        JobReport jobReport = new JobReport()
                .withBucket(reportBucketName)
                .withPrefix("reports")
                .withFormat("Report_CSV_20180820")
                .withEnabled(true)
                .withReportScope("AllTasks");

        AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(US_WEST_2)
                .build();

        s3ControlClient.createJob(new CreateJobRequest()
                .withAccountId(accountId)
                .withOperation(jobOperation)
                .withManifest(manifest)
                .withReport(jobReport)
                .withPriority(42)
                .withRoleArn(iamRoleArn)
                .withClientRequestToken(uuid)
                .withDescription("job description")
                .withConfirmationRequired(false)
        );

    } catch (AmazonServiceException e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it and returned an error response.
        e.printStackTrace();
    } catch (SdkClientException e) {
        // Amazon S3 couldn't be contacted for a response, or the client
        // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

Generate manifest

以下示例说明如何创建 S3 批量操作 s3PutObjectCopy 任务，以便根据对象筛选标准（包括创建日期、密钥名称和大小）自动生成清单。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

Example

```
        package aws.example.s3control;

    import com.amazonaws.AmazonServiceException;
    import com.amazonaws.SdkClientException;
    import com.amazonaws.auth.profile.ProfileCredentialsProvider;
    import com.amazonaws.services.s3control.AWSS3Control;
    import com.amazonaws.services.s3control.AWSS3ControlClient;
    import com.amazonaws.services.s3control.model.CreateJobRequest;
    import com.amazonaws.services.s3control.model.CreateJobResult;
    import com.amazonaws.services.s3control.model.JobManifestGenerator;
    import com.amazonaws.services.s3control.model.JobManifestGeneratorFilter;
    import com.amazonaws.services.s3control.model.JobOperation;
    import com.amazonaws.services.s3control.model.JobReport;
    import com.amazonaws.services.s3control.model.KeyNameConstraint;
    import com.amazonaws.services.s3control.model.S3JobManifestGenerator;
    import com.amazonaws.services.s3control.model.S3ManifestOutputLocation;
    import com.amazonaws.services.s3control.model.S3SetObjectTaggingOperation;
    import com.amazonaws.services.s3control.model.S3Tag;

    import java.time.Instant;
    import java.util.Date;
    import java.util.UUID;
    import java.util.ArrayList;

    import static com.amazonaws.regions.Regions.US_WEST_2;

    public class test {
        public static void main(String[] args) {
            String accountId = "012345678901";
            String iamRoleArn = "arn:aws:iam::012345678901:role/ROLE";
```

```
            String sourceBucketName = "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET";
            String reportBucketName = "arn:aws:s3:::DOC-EXAMPLE-REPORT-BUCKET";
            String manifestOutputBucketName = "arn:aws:s3:::DOC-EXAMPLE-MANIFEST-
OUTPUT-BUCKET";
            String uuid = UUID.randomUUID().toString();
            long minimumObjectSize = 100L;

            ArrayList<S3Tag> tagSet = new ArrayList<>();
            tagSet.add(new S3Tag().withKey("keyOne").withValue("ValueOne"));

            ArrayList<String> prefixes = new ArrayList<>();
            prefixes.add("s3KeyStartsWith");

            try {
                JobOperation jobOperation = new JobOperation()
                        .withS3PutObjectTagging(new S3SetObjectTaggingOperation()
                                .withTagSet(tagSet)
                        );
                S3ManifestOutputLocation manifestOutputLocation = new
 S3ManifestOutputLocation()
                        .withBucket(manifestOutputBucketName)
                        .withManifestPrefix("manifests")
                        .withExpectedManifestBucketOwner(accountId)
                        .withManifestFormat("S3InventoryReport_CSV_20211130");

                JobManifestGeneratorFilter jobManifestGeneratorFilter = new
 JobManifestGeneratorFilter()
                        .withEligibleForReplication(true)
                        .withKeyNameConstraint(
                                new KeyNameConstraint()
                                        .withMatchAnyPrefix(prefixes))
                        .withCreatedBefore(Date.from(Instant.now()))
                        .withObjectSizeGreaterThanBytes(minimumObjectSize);

                S3JobManifestGenerator s3JobManifestGenerator = new
 S3JobManifestGenerator()
                        .withEnableManifestOutput(true)
                        .withManifestOutputLocation(manifestOutputLocation)
                        .withFilter(jobManifestGeneratorFilter)
                        .withSourceBucket(sourceBucketName);

                JobManifestGenerator jobManifestGenerator = new
 JobManifestGenerator()
                        .withS3JobManifestGenerator(s3JobManifestGenerator);
```

```
                JobReport jobReport = new JobReport()
                        .withBucket(reportBucketName)
                        .withPrefix("reports")
                        .withFormat("Report_CSV_20180820")
                        .withEnabled(true)
                        .withReportScope("AllTasks");

                AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                        .withCredentials(new ProfileCredentialsProvider())
                        .withRegion(US_WEST_2)
                        .build();

                CreateJobResult createJobResult = s3ControlClient.createJob(new
CreateJobRequest()
                        .withAccountId(accountId)
                        .withOperation(jobOperation)
                        .withManifestGenerator(jobManifestGenerator)
                        .withReport(jobReport)
                        .withPriority(42)
                        .withRoleArn(iamRoleArn)
                        .withClientRequestToken(uuid)
                        .withDescription("job description")
                        .withConfirmationRequired(true)
                );

                System.out.println("Created job " + createJobResult.getJobId());

            } catch (AmazonServiceException e) {
                // The call was transmitted successfully, but Amazon S3 couldn't
process
                // it and returned an error response.
                e.printStackTrace();
            } catch (SdkClientException e) {
                // Amazon S3 couldn't be contacted for a response, or the client
                // couldn't parse the response from Amazon S3.
                e.printStackTrace();
            }
        }
    }
```

使用 REST API

您可以使用 REST API 创建批量操作任务。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CreateJob。

任务响应

如果 `CreateJob` 请求成功，Amazon S3 将返回一个任务 ID。任务 ID 是 Amazon S3 自动生成的唯一标识符，以便于您标识批量操作任务并监控其状态。

通过 AWS CLI、AWS SDK 或 REST API 创建任务时，您可以设置 S3 批量操作以开始自动处理任务。任务在准备就绪后立即开始运行，而不是等待后面较高优先级的任务。

当您通过 Amazon S3 控制台创建任务时，在批量操作开始处理该任务之前，您必须查看任务详细信息并确认希望运行它。如果任务保持暂停状态超过 30 天，则它将失败。

S3 分批操作支持的操作

S3 分批操作支持多种不同的操作。此部分中的主题介绍了所有这些操作。

复制对象

Copy 操作会复制清单中指定的每个对象。您可以将对象复制到相同 AWS 区域中的存储桶或不同区域中的存储桶。S3 分批操作支持 Amazon S3 所提供用于复制对象的大部分选项。这些选项包括设置对象元数据、设置权限以及更改对象的存储类。

您可以使用复制操作复制现有的未加密对象，并将其写入同一存储桶作为加密对象。有关更多信息，请参阅使用 Amazon S3 分批操作加密对象。

复制对象时，可以更改用于计算对象校验和的校验和算法。如果对象没有额外的已计算校验和，您还可以通过指定 Amazon S3 要使用的校验和算法来添加一个校验和。有关更多信息，请参阅检查对象完整性。

有关复制 Amazon S3 中的对象以及必需参数和可选参数的更多信息，请参阅本指南中的 复制、移动和重命名对象 和《Amazon Simple Storage Service API 参考》中的 CopyObject。

限制和局限性

- 所有源对象必须位于一个存储桶中。
- 所有目标对象必须位于一个存储桶中。
- 您必须具有源存储桶的读取权限和目标存储桶的写入权限。

- 复制的对象大小最多为 5 GB。
- 如果您尝试将对象从 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 类复制到 S3 Standard 存储类，则需要先还原这些对象。有关更多信息，请参阅恢复已归档的对象。
- 复制任务必须在目标区域（您要将对象复制到的区域）中创建。
- 除了 ETag 的条件检查和使用客户提供的加密密钥的服务器端加密 (SSE-C) 之外，所有复制选项都受支持。
- 如果存储桶为非版本化，您将覆盖具有相同键名的对象。
- 不必以对象在清单中出现的相同顺序复制这些对象。对于受版本控制的存储桶，如果保留当前/非当前版本顺序很重要，您应首先复制所有非当前版本。然后，在第一个任务完成后，在后续任务中复制当前版本。
- 不支持将对象复制到低冗余存储 (RRS) 类。

使用 S3 分批操作复制对象

您可以使用 S3 分批操作创建 PUT 复制任务以复制同一账户内的对象或将对象复制到不同的目标账户。以下部分包含有关如何存储和使用不同账户中的清单的示例。在第一个部分中，您可以使用 Amazon S3 清单将清单报告传输到目标账户以在任务创建期间使用，或者可以在源或目标账户中使用逗号分隔值 (CSV) 清单，如第二个示例所示。第三个示例说明如何使用 Copy 操作激活现有对象上的 S3 Bucket 密钥加密。

复制操作示例

- 使用发送到目标账户的清单报告来跨 AWS 账户 复制对象
- 使用 CSV 清单跨 AWS 账户 复制对象
- 使用 S3 分批操作加密具有 S3 Bucket 密钥的对象

使用发送到目标账户的清单报告来跨 AWS 账户 复制对象

您可以使用 Amazon S3 清单创建清单报告，然后使用该报告创建要使用 S3 分批操作复制的对象列表。有关在源账户或目标账户中使用 CSV 清单的信息，请参阅 the section called “使用 CSV 清单跨 AWS 账户 复制对象”。

Amazon S3 清单生成桶中的对象的清单。生成的列表将发布到输出文件。进行清点的桶称为源桶，其中存储清单报告文件的桶称为目标桶。

Amazon S3 清单报告可配置为传输到另一个 AWS 账户。这将在目标账户中创建任务时允许 S3 分批操作读取清单报告。

有关 Amazon S3 清单源桶和目标桶的更多信息，请参阅 源存储桶和目标存储桶。

设置清单的最简单方法是使用 AWS Management Console，不过您也可以使用 REST API、AWS Command Line Interface (AWS CLI) 或 AWS 开发工具包。

以下控制台过程包含用于设置 S3 分批操作任务的权限的概要步骤。在此过程中，您将对象从源账户复制到目标账户，并将清单报告存储在目标账户中。

为由不同账户拥有的源桶和目标桶设置 Amazon S3 清单

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 选择要将清单报告存储在其中的目标桶。

   确定用于存储清单报告的目标清单桶。在此过程中，目标账户 是同时拥有目标清单桶和对象要复制到的桶的账户。
3. 配置一个清单以列出源桶中的对象并将该列表发布到目标清单桶。

   为源桶配置清单列表。执行此操作时，指定要将该列表存储在其中的目标桶。源桶的清单报告将发布到目标桶。在此过程中，源账户 是拥有源桶的账户。

   有关如何使用控制台配置清单或如何加密清单列表文件的信息，请参阅配置 Amazon S3 清单。

   为输出格式选择 CSV。

   输入目标桶的信息时，请选择 Buckets in another account (另一个账户中的桶)。然后，输入目标清单桶的名称。或者，可以输入目标账户的账户 ID。

   在保存清单配置后，控制台会显示类似于以下内容的一条消息：

   Amazon S3 无法在目标桶上创建桶策略。要求目标桶拥有者添加以下桶策略，以允许 Amazon S3 在该桶中放置数据。

   然后，控制台会显示可用于目标桶的桶策略。
4. 复制显示在控制台上的目标桶策略。
5. 在目标账户中，将复制的桶策略添加到存储清单报告所在的目标清单桶。
6. 在目标账户中创建一个基于 S3 分批操作信任策略的角色。有关信任策略的更多信息，请参阅信任策略。

   有关创建角色的更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

输入角色的名称（示例角色使用名称 BatchOperationsDestinationRoleCOPY）。选择 S3 服务，然后选择 S3 bucket Batch Operations (S3 桶分批操作) 使用案例，这会将信任策略应用于该角色。

然后选择 Create policy (创建策略) 以将以下策略附加到该角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::ObjectDestinationBucket/*",
        "arn:aws:s3:::ObjectSourceBucket/*",
        "arn:aws:s3:::ObjectDestinationManifestBucket/*"
      ]
    }
  ]
}
```

该角色使用策略授予 batchoperations.s3.amazonaws.com 权限来读取目标桶中的清单。它还授予对源对象桶中的 GET 对象、访问控制列表 (ACL)、标签和版本的权限。此外，还授予对目标对象桶中的 PUT 对象、ACL、标签和版本的权限。

7. 在源账户中，为源桶创建一个桶策略，该策略向您在上一步中创建的角色授予源桶中的 GET 对象、ACL、标签和版本的权限。此步骤允许 S3 分批操作通过可信角色获取源桶中的对象。

以下是源账户的桶策略的示例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceObjectCOPY",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::DestinationAccountNumber:role/
BatchOperationsDestinationRoleCOPY"
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
    }
  ]
}
```

8. 在清单报告可用后，在目标账户中创建一个 S3 分批操作 PUT 对象复制任务，同时从目标清单桶中选择清单报告。您需要在目标账户中创建的角色的 ARN。

   有关创建任务的一般信息，请参阅 创建 S3 批量操作任务。

   有关使用控制台创建任务的信息，请参阅 创建 S3 批量操作任务。

使用 CSV 清单跨 AWS 账户 复制对象

可以使用存储在源账户中的 CSV 清单，通过 S3 批量操作跨 AWS 账户复制对象。要使用 S3 清单报告作为清单，请参阅the section called “使用清单报告跨 AWS 账户 复制对象”。

有关清单文件的 CSV 格式的示例，请参阅the section called “创建清单文件”。

以下过程显示了当使用 S3 批量操作任务通过存储在源账户中的 CSV 清单文件将对象从源账户复制到目标账户时，如何设置权限。

使用 CSV 清单跨 AWS 账户复制对象

1. 在目标账户中创建一个基于 S3 分批操作信任策略的角色。在此过程中，目标账户是对象要复制到的账户。

   有关信任策略的更多信息，请参阅信任策略。

   有关创建角色的更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

   如果您使用控制台来创建角色，请输入角色的名称（示例角色使用名称 `BatchOperationsDestinationRoleCOPY`）。选择 S3 服务，然后选择 S3 bucket Batch Operations（S3 桶分批操作）使用案例，这会将信任策略应用于该角色。

   然后，选择创建策略来将以下策略附加到该角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::ObjectDestinationBucket/*",
        "arn:aws:s3:::ObjectSourceBucket/*",
        "arn:aws:s3:::ObjectSourceManifestBucket/*"
      ]
    }
  ]
}
```

使用策略，该角色将授予 batchoperations.s3.amazonaws.com 权限以读取源清单存储桶中的清单。该角色授予对源对象存储桶中的 GET 对象、访问控制列表（ACL）、标签和版本的权限。它还授予对目标对象存储桶中的 PUT 对象、ACL、标签和版本的权限。

2. 在源账户中，为包含清单的存储桶创建存储桶策略，来向您在上一步中创建的角色授予对源清单存储桶中 GET 对象和版本的权限。

   此步骤可让 S3 批量操作通过使用可信角色来读取清单。将存储桶策略应用于包含清单的存储桶。

   以下是要应用于源清单存储桶的存储桶策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceManfiestRead",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::DestinationAccountNumber:user/ConsoleUserCreatingJob",

 "arn:aws:iam::DestinationAccountNumber:role/BatchOperationsDestinationRoleCOPY"
        ]
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceManifestBucket/*"
    }
  ]
}
```

   此策略还授予权限来允许要在目标账户中创建任务的控制台用户通过相同的存储桶策略具有源清单存储桶中的相同权限。

3. 在源账户中，为源存储桶创建存储桶策略，该策略向您创建的角色授予对源对象存储桶中 GET 对象、ACL、标签和版本的权限。然后，S3 分批操作可以通过可信角色获取源存储桶中的对象。

   以下是包含源对象的存储桶的存储桶策略示例：

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceObjectCOPY",
      "Effect": "Allow",
      "Principal": {
        "AWS":
 "arn:aws:iam::DestinationAccountNumber:role/BatchOperationsDestinationRoleCOPY"
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::ObjectSourceBucket/*"
    }
  ]
}
```

4. 在目标账户中创建 S3 分批操作任务。您需要您在目标账户中创建的角色的 Amazon Resource Name (ARN)。有关创建作业的更多信息，请参阅创建 S3 批量操作任务。

使用 S3 分批操作加密具有 S3 Bucket 密钥的对象

在本节中，您可以使用 Amazon S3 分批操作 Copy 操作来识别和激活现有对象上的 S3 Bucket 密钥加密。有关 S3 Bucket 密钥的更多信息，请参阅 使用 Amazon S3 存储桶密钥降低 SSE-KMS 的成本 和 将存储桶配置为将 S3 存储桶密钥与 SSE-KMS 结合使用于新对象。

本示例中涵盖的主题包括：

主题

- 先决条件
- 第 1 步：使用 Amazon S3 清单获取对象列表
- 第 2 步：使用 S3 Select 筛选对象列表
- 第 3 步：设置并运行 S3 分批操作任务
- Summary

先决条件

要按照此过程中的步骤执行操作，您必须具有 AWS 账户和至少一个 S3 桶来存放工作文件和加密的结果。您可能还会发现许多有用的现有 S3 分批操作文档，包括以下主题：

- S3 批量操作基础知识
- 创建 S3 批量操作任务
- S3 分批操作支持的操作
- 管理 S3 分批操作任务

第 1 步：使用 Amazon S3 清单获取对象列表

要开始操作，请确定包含要加密对象的 S3 Bucket，并获取其内容列表。Amazon S3 清单报告是完成此操作的最方便且最经济的方式。报告提供桶对象列表以及相关元数据。源桶指的是进行清点的桶，目标桶指的是您存储清单报告文件的桶。有关 Amazon S3 清单源桶和目标桶的更多信息，请参阅 Amazon S3 清单。

设置清单最简单的方法是使用 AWS Management Console。但您也可以使用 REST API、AWS Command Line Interface (AWS CLI) 或 AWS 开发工具包。执行以下步骤之前，请务必登录控制台，并通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。如果遇到权限拒绝错误，请在目标桶中添加桶策略。有关更多信息，请参阅向 S3 清单和 S3 分析功能授予权限。

使用 S3 清单获取对象列表

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，选择 Buckets（桶），然后选择包含要加密对象的桶。
3. 在 Management (管理) 选项卡导航至 Inventory configurations (清单配置) 部分，然后选择 Create inventory configuration (创建清单配置)。
4. 为您的新清单命名，输入目标 S3 Bucket 的名称，并可选择为 Amazon S3 创建目标前缀，以便在该桶中分配对象。
5. 为 Output format (输出格式) 选择 CSV。
6. （可选）在其他字段 – 可选部分中，选择加密以及您感兴趣的任何其他报告字段。将报告发送频率设置为 Daily（每天），以便更快将第一个报告发送到您的桶。
7. 选择 Create (创建) 以保存您的配置。

Amazon S3 可能需要长达 48 小时才能交付第一个报告，所以当第一个报告到达时请予以查看。收到第一个报告后，请进入下一部分以筛选 S3 清单报告的内容。如果您不再希望接收此桶的清单报告，请删除您的 S3 清单配置。否则，S3 会按每日或每周计划提供报告。

清单列表并非所有对象的单个时间点视图。清单列表是桶项的滚动快照，最终是一致的（例如，列表可能不包含最近添加或删除的对象）。当您使用静态对象或两天乃至更多天之前创建的对象集时，将 S3 清单和 S3 分批操作结合使用效果最佳。要处理最新数据，请使用 ListObjectsV2（GET 桶）API 操作来手动构建对象列表。如有需要，请在接下来的几天内重复该流程，或直到您的清单报告显示所有密钥的理想状态。

第 2 步：使用 S3 Select 筛选对象列表

收到 S3 清单报告后，您可以筛选报告的内容，以仅列出未使用 S3 桶密钥加密的对象。如果您希望使用 S3 桶密钥加密桶的所有对象，则可以忽略此步骤。但是，在此阶段过滤 S3 清单报告可节省重新加密先前已加密对象的时间和费用。

尽管以下步骤介绍的是如何使用 Amazon S3 Select，您也可以使用 Amazon Athena。要决定使用哪个工具，请查看 S3 清单报告的 `manifest.json` 文件。此文件列出了与该报告关联的数据文件的数量。如果数量较大，请使用 Amazon Athena，因为其运行在多个 S3 对象之间，而 S3 Select 一次仅适用一个对象。有关将 Amazon S3 和 Athena 结合使用的更多信息，请参阅 使用 Amazon Athena 查询 Amazon S3 清单 以及博客文章使用 Amazon S3 分批操作加密对象中的使用 Athena。

使用 S3 Select 筛选 S3 清单报告

1. 打开清单报告中的 `manifest.json` 文件，然后查看 JSON 中的 `fileSchema` 部分。这将通知您的数据运行查询。

   以下 JSON 是一个示例 `manifest.json` 文件，用于查看已启用版本控制的桶上的 CSV 格式清单。根据您配置库存报告的方式，您的清单可能会有所不同。

```
 {
   "sourceBucket": "batchoperationsdemo",
   "destinationBucket": "arn:aws:s3:::testbucket",
   "version": "2021-05-22",
   "creationTimestamp": "1558656000000",
   "fileFormat": "CSV",
   "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
BucketKeyStatus",
   "files": [
     {
```

```
        "key": "demoinv/batchoperationsdemo/DemoInventory/data/009a40e4-
f053-4c16-8c75-6100f8892202.csv.gz",
        "size": 72691,
        "MD5checksum": "c24c831717a099f0ebe4a9d1c5d3935c"
      }
    ]
  }
```

如果桶未激活版本控制，或者您选择运行最新版本报告，则 `fileSchema` 是 `Bucket`、`Key` 和 `BucketKeyStatus`。

如果激活了版本控制，根据您设置清单报告的方式，`fileSchema` 可能包括：`Bucket`、`Key`、`VersionId`、`IsLatest`、`IsDeleteMarker`、`BucketKeyStatus`。因此，当您运行查询时，请注意第 1、2、3 和 6 栏。

除了搜索依据字段（即 `BucketKeyStatus`）外，S3 批量操作还需要输入桶、密钥和版本 ID 以执行任务。您不需要版本 ID 字段，但在您对受版本控制的桶进行操作时，该字段有助于指定桶。有关更多信息，请参阅使用启用版本控制的存储桶中的对象。

2. 找到清单报告的数据文件。`manifest.json` 对象列出了文件夹下的数据文件。
3. 在 S3 控制台中找到并选择数据文件后，选择 Actions (操作)，然后选择 Query with S3 Select (使用 S3 Select 进行查询)。
4. 保留预设 CSV、逗号 和 GZIP 字段，然后选择 Next (下一步)。
5. 如需在继续操作之前检查清单报告格式，请选择 Show file preview (显示文件预览)。
6. 在 SQL 表达式字段中输入要引用的列，然后选择 Run SQL（运行 SQL）。以下表达式为未配置 S3 桶密钥的所有对象返回列 1-3。

```
select s._1, s._2, s._3 from s3object s where s._6 = 'DISABLED'
```

以下是示例结果。

```
      batchoperationsdemo,0100059%7Ethumb.jpg,lsrtIxksLu0R0ZkYPL.LhgD5caTYn6vu
      batchoperationsdemo,0100074%7Ethumb.jpg,sd2M60g6Fdazoi6D5kNARIE7KzUibmHR
      batchoperationsdemo,0100075%7Ethumb.jpg,TLYESLnl1mXD5c4BwiOIinqFrktddkoL
      batchoperationsdemo,0200147%7Ethumb.jpg,amufzfMi_fEw0Rs99rxR_HrDFlE.l3Y0
      batchoperationsdemo,0301420%7Ethumb.jpg,9qGU2SEscL.C.c_sK89trmXYIwooABSh
      batchoperationsdemo,0401524%7Ethumb.jpg,ORnEWNuB1QhHrrYAGFsZhbyvEYJ3DUor
      batchoperationsdemo,200907200065HQ
%7Ethumb.jpg,d8LgvIVjbDR5mUVwW6pu9ahTfReyn5V4
```

```
      batchoperationsdemo,200907200076HQ
%7Ethumb.jpg,XUT25d7.gK40u_GmnupdaZg3BVx2jN40
      batchoperationsdemo,201103190002HQ
%7Ethumb.jpg,z.2sVRh0myqVi0BuIrngWlsRPQdb7qOS
```

7. 下载结果，将其保存为 CSV 格式，然后将其作为 S3 分批操作任务的对象列表上传至 Amazon S3。
8. 如果您有多个清单文件，也请对它们运行 Query with S3 Select (使用 S3 Select 进行查询)。根据结果的大小，您可以合并列表并运行单个 S3 分批操作任务，或将每个列表作为单独任务运行。

   当您决定要运行的任务数量时，请考虑运行每个 S3 分批操作任务的价格。

第 3 步：设置并运行 S3 分批操作任务

现在，您已经有了 S3 对象的筛选 CSV 列表，您可以开始 S3 分批操作任务以使用 S3 Bucket 密钥加密对象。

任务指提供的对象列表（清单）、执行的操作以及指定的参数的统称。加密此对象集的最简单方法是使用 PUT 复制操作并指定与清单中列出对象相同的目标前缀。这会覆盖未受版本控制的桶中的现有对象，或在启用版本控制的情况下，创建更新的加密版本对象。

作为复制对象的一部分，指定 Amazon S3 应使用 SSE-KMS 加密和 S3 来为对象进行加密。此任务会复制这些对象，因此，所有对象在完成时都显示更新后的创建日期，无论您最初是何时将其添加到 S3 的。另外，在 S3 分批操作任务中指定对象集的其他属性，包括对象标记和存储类。

分步

- 设置 IAM 策略
- 设置分批操作 IAM 角色
- 为现有桶启用 S3 Bucket 密钥
- 创建分批操作任务
- 运行您的分批操作任务
- 需知信息

设置 IAM 策略

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中选择 Policies (策略)，然后选择 Create Policy (创建策略)。

3. 选择 JSON 选项卡。选择 Edit Policy (编辑策略)，然后添加出现在以下代码块中的示例 IAM 策略。

   将策略示例复制到 IAM 控制台后，请替换以下内容：

   a. 将 *SOURCE_BUCKET_FOR_COPY* 替换为您的源桶的名称。

   b. 将 *DESTINATION_BUCKET_FOR_COPY* 替换为您的目标桶的名称。

   c. 将 *MANIFEST_KEY* 替换为清单对象的名称。

   d. 将 *REPORT_BUCKET* 替换为您想用于保存报告的桶的名称。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CopyObjectsToEncrypt",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectTagging",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectVersionAcl",
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::SOURCE_BUCKET_FOR_COPY/*",
        "arn:aws:s3:::DESTINATION_BUCKET_FOR_COPY/*"
      ]
    },
    {
      "Sid": "ReadManifest",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
```

```
        "Resource": "arn:aws:s3:::MANIFEST_KEY"
      },
      {
        "Sid": "WriteReport",
        "Effect": "Allow",
        "Action": [
          "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::REPORT_BUCKET/*"
      }
    ]
}
```

4. 选择 Next: Tags (下一步：标签)。
5. 添加您想要的所有标签（可选），然后选择 Next: Review (下一步：审核)。
6. 添加策略名称，可添加描述，然后选择 Create Policy (创建策略)。
7. 选择 Review policy (查看策略) 和 Save changes (保存更改)。
8. 您的 S3 分批操作策略现已完成，控制台会将您返回到 IAM Policies (策略) 页面。筛选策略名称，选择策略名称左侧的按钮，选择 Policy actions (策略操作)，然后选择 Attach (附加)。

   如需将新创建的策略附加到 IAM 角色，请在账户中选择相应的用户、组或角色，然后选择 Attach policy (附加策略)。系统会带您返回 IAM 控制台。

设置分批操作 IAM 角色

1. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
2. 选择 AWS 服务、S3 和 S3 批量操作。然后选择 Next: Permissions (下一步：权限)。
3. 输入您刚刚创建的 IAM 策略的名称。根据出现的策略名称选中相应名称，然后选择 Next: Tags (下一步：标签)。
4. （可选）为本次练习添加标签或将密钥和值字段保留为空。选择 Next: Review (下一步：审核)。
5. 输入角色名称，然后接受默认说明或添加您自己的说明。选择 Create role (创建角色)。
6. 确保创建任务的用户具有以下示例中的权限。

   将 *{ACCOUNT-ID}* 替换为您的 AWS 账户 ID，并将 *{IAM_ROLE_NAME}* 替换为您计划应用于稍后将在批量操作任务创建步骤中创建的 IAM 角色的名称。有关更多信息，请参阅授予 Amazon S3 分批操作的权限。

```
            {
```

```
            "Sid": "AddIamPermissions",
            "Effect": "Allow",
            "Action": [
            "iam:GetRole",
            "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::ACCOUNT-ID:role/IAM_ROLE_NAME"
            }
```

为现有桶启用 S3 Bucket 密钥

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在桶列表中，选择要为其启用 S3 桶密钥的桶。
3. 选择 Properties (属性)。
4. 在 Default encryption (默认加密) 下，请选择 Edit (编辑)。
5. 在加密类型下，您可以在 Amazon S3 托管式密钥（SSE-S3）和 AWS Key Management Service 密钥（SSE-KMS）之间进行选择。
6. 如果您选择了 AWS Key Management Service 密钥（SSE-KMS），则在 AWS KMS key 之下，您可以通过以下选项之一指定 AWS KMS 密钥。
   - 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS 密钥中进行选择。从可用密钥的列表中，选择与桶位于同一区域的对称加密 KMS 密钥。AWS 托管式密钥（aws/s3）和您的客户自主管理型密钥都显示在列表中。
   - 要输入 KMS 密钥 ARN，请选择输入 AWS KMS 密钥 ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
   - 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。
7. 在 Bucket Key（桶密钥）中，选择 Enable（启用），然后选择 Save changes（保存更改）。

现在，S3 桶密钥已在桶级别开启，原定设置情况下，上传、修改或复制到此桶中的对象将承继此加密配置。这包括使用 Amazon S3 批量操作复制的对象。

创建分批操作任务

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，选择 Batch Operations (分批操作)，然后选择 Create Job (创建任务)。
3. 选择您存储对象的 Region (区域)，然后选择 CSV 作为清单类型。

4. 输入路径或导航至您之前从 S3 Select（或 Athena）结果创建的 CSV 清单文件。如果您的清单包含版本 ID，请选中该框。选择 Next (下一步)。
5. 选择 Copy 操作，然后选择复制目标桶。您可以保持禁用服务器端加密。只要桶目标启用了 S3 桶密钥，复制操作将在目标桶上应用 S3 桶密钥。
6. （可选）根据需要选择存储类和其他参数。您在此步骤中指定的参数将应用于清单中列出的对象执行的所有操作。选择下一步。
7. 要配置服务器端加密，请执行以下步骤：

   a. 在服务器端加密下，选择以下选项之一：

      - 要在将对象存储在 Amazon S3 中时保留适用于对象原定设置服务器端加密的桶设置，请选择不要指定加密密钥。只要桶目标启用了 S3 桶密钥，复制操作就将在目标桶上应用 S3 桶密钥。

        > **Note**
        >
        > 如果指定的目标的桶策略要求在将对象存储到 Amazon S3 之前对其进行加密，则必须指定加密密钥。否则，将对象复制到目标将失败。

      - 要在将对象存储到 Amazon S3 之前对其进行加密，请选择指定加密密钥。

   b. 在加密设置下，如果您选择指定加密密钥，则必须选择使用用于原定设置加密的目标桶设置或覆盖用于原定设置加密的目标桶设置。

   c. 如果您选择覆盖用于原定设置加密的目标桶设置，则必须配置以下加密设置。

      i. 在加密类型下，您必须选择 Amazon S3 托管式密钥（SSE-S3）或 AWS Key Management Service 密钥（SSE-KMS）。SSE-S3 使用最强的数据块密码之一 [即 256 位高级加密标准（AES-256）] 来加密每个对象。SSE-KMS 为您提供了对密钥的更多控制。有关更多信息，请参阅 使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）和 使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）。

      ii. 如果您选择 AWS Key Management Service 密钥（SSE-KMS），则在 AWS KMS key 之下，您可以通过以下选项之一指定 AWS KMS key。

         - 要从可用 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 中进行选择，然后选择与桶位于同一区域的对称加密 KMS 密钥。AWS 托管式密钥（aws/s3）和您的客户自主管理型密钥都显示在列表中。
         - 要输入 KMS 密钥 ARN，请选择输入 AWS KMS 密钥 ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。

- 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

iii. 在 Bucket Key（存储桶密钥）下，请选择 Enable（启用）。复制操作将在目标桶应用 S3 桶密钥。

8. 提供任务说明（或保留默认值），设置其优先级，选择报告类型，然后指定 Path to completion report destination (完成报告目标的路径)。
9. 在 Permissions (权限) 部分，请确保选择您之前定义的分批操作 IAM 角色。选择 Next (下一步)。
10. 在 Review (审核) 中验证设置。如果需要进行更改，请选择 Previous (上一步)。确认分批操作设置后，选择 Create job (创建任务)。

    有关更多信息，请参阅 创建 S3 批量操作任务。

运行您的分批操作任务

安装向导将自动将您返回 Amazon S3 控制台的 S3 分批操作部分。随着 S3 开始此过程，您的新任务将从 New (新建) 状态变为 Preparing (准备中) 状态。在 Preparing (准备中) 状态下，S3 会读取任务清单，检查是否存在错误，并计算对象数量。

1. 在 Amazon S3 控制台中选择刷新按钮以检查进度。根据清单的大小，读取可能需要几分钟或几小时。
2. S3 读取任务清单后，任务将转移至 Awaiting your confirmation (待确认) 状态。选择任务 ID 左侧的选项按钮，然后选择 Run job (运行任务)。
3. 检查任务设置，然后选择右下角的 Run job (运行任务)。

   任务开始运行后，您可以选择刷新按钮通过控制台控制面板视图或选择特定任务来检查进度。
4. 任务完成后，您可以查看 Successful (成功) 和 Failed (失败) 对象数量以确认所有任务均按预期执行。如果您启用了任务报告，请检查任务报告，了解失败操作的确切原因。

   还可以使用 AWS CLI、AWS SDK 或 Amazon S3 REST API 来执行这些步骤。有关追踪任务状态和完成报告的更多信息，请参阅 跟踪任务状态和完成报告。

需知信息

使用 S3 批量操作通过 S3 桶密钥加密对象时，请考虑以下问题：

- 除了与 S3 批量操作代表您执行的操作关联的任何费用之外（包括数据传输、请求等费用），您还需要为 S3 批量操作任务、对象和请求付费。有关更多信息，请参阅 Amazon S3 定价。

- 如果您使用受版本控制的桶，则执行的每个 S3 分批操作任务都会创建对象的新加密版本。另外，无需配置 S3 桶密钥即可维护以前的版本。要删除旧版本，请为非当前版本设置 S3 生命周期过期策略，如 生命周期配置元素。
- 复制操作会创建具有新创建日期的新对象，这会影响生命周期操作（如归档）。如果您复制桶中的所有对象，所有新副本均具有相同或相似的创建日期。要进一步识别这些对象并为各种数据子集创建不同的生命周期规则，请考虑使用对象标签。

Summary

在本节中，您将对现有对象进行整理，筛选出已加密的数据。然后，通过 S3 批量操作将现有数据复制到 S3 密钥已激活的桶，对未加密的对象应用 S3 桶密钥功能。这个过程可以为您节省时间和金钱，同时允许您完成诸如加密所有现有对象等操作。

有关 S3 分批操作的更多信息，请参阅 对 Amazon S3 对象执行大规模批量操作。

有关说明使用 AWS CLI 和 AWS SDK for Java 对标签进行复制操作的示例，请参阅 使用用于标记的任务标签创建分批操作任务。

调用 AWS Lambda 函数

调用 AWS Lambda 函数启动 AWS Lambda 函数，来对清单中列出的对象执行自定义操作。本部分介绍如何创建 Lambda 函数来与 S3 批量操作结合使用，以及如何创建任务来调用该函数。S3 批量操作任务使用 `LambdaInvoke` 操作对清单中列出的每个对象运行 Lambda 函数。

您可以通过使用 AWS Management Console、AWS Command Line Interface（AWS CLI）、AWS SDK 或 REST API，来使用适用于 Lambda 的 S3 批量操作。有关使用 Lambda 的更多信息，请参阅《AWS Lambda 开发人员指南》中的入门AWS Lambda。

以下部分说明如何开始将 S3 批量操作与 Lambda 结合使用。

主题

- 将 Lambda 与 Amazon S3 批量操作结合使用
- 创建与 S3 批量操作一起使用的 Lambda 函数
- 创建调用 Lambda 函数的 S3 批量操作任务
- 在 Lambda 清单中提供任务级信息
- 从 S3 批量操作教程中学习

将 Lambda 与 Amazon S3 批量操作结合使用

将 S3 批量操作与 AWS Lambda 结合使用时，您必须创建专门用于 S3 批量操作的新 Lambda 函数。您无法在 S3 批量操作中重复使用基于事件的现有 Amazon S3 函数。事件函数只能接收消息；不返回消息。与 S3 批量操作结合使用的 Lambda 函数必须接受并返回消息。有关将 Lambda 与 Amazon S3 事件结合使用的更多信息，请参阅《AWS Lambda 开发人员指南》中的将 AWS Lambda 与 Amazon S3 结合使用。

创建调用 Lambda 函数的 S3 批量操作任务。此任务在清单中列出的所有对象上运行相同的 Lambda 函数。在处理清单中的对象时，您可以控制要使用的 Lambda 函数版本。S3 批量操作支持非限定的 Amazon Resource Name（ARN）、别名和特定版本。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的AWS Lambda 版本控制简介。

如果您提供的 S3 批量操作任务包含使用别名或 `$LATEST` 限定符的函数 ARN，并更新它们各自所指向的版本，S3 批量操作开始调用 Lambda 函数的新版本。如果您希望通过大型任务更新此方法的功能部分，这会非常有用。如果不您希望 S3 批量操作更改所使用的版本，请在创建任务时在 `FunctionARN` 参数中提供特定版本。

将 Lambda 和 Amazon S3 批量操作用于目录存储桶

目录存储桶是一种 Amazon S3 存储桶，专为需要一致的个位数毫秒延迟的工作负载或性能至关重要的应用程序而设计。有关更多信息，请参阅目录存储桶。

使用 Amazon S3 批量操作来调用处理目录存储桶的 Lambda 函数有特殊要求。例如，您必须使用更新后的 JSON 架构来构建 Lambda 请求，并在创建任务时指定 `InvocationSchemaVersion` **`2.0`**。此更新后的架构允许您为 `UserArguments` 指定可选的键值对，您可以使用这些键值对修改现有 Lambda 函数的某些参数。有关更多信息，请参阅 AWS 存储博客中的 Automate object processing in Amazon S3 directory buckets with S3 Batch Operations and AWS Lambda。

响应和结果代码

S3 批量操作使用一个或多个键调用 Lambda 函数，每个键都有一个与之关联的 `TaskID`。S3 批量操作需要 Lambda 函数提供每个键的结果代码。对于请求中发送的任何任务 ID，如果没有为它们返回每个键的结果代码，则将从 `treatMissingKeysAs` 字段中获得结果代码。`treatMissingKeysAs` 是可选的请求字段，默认为 `TemporaryFailure`。下表包含 `treatMissingKeysAs` 字段的其它可能的结果代码和值。

响应代码	描述
`Succeeded`	任务正常完成。如果您请求了任务完成报告，报告中将包含任务的结果字符串。
`TemporaryFailure`	任务暂时失败，将在任务完成前重新启动。忽略结果字符串。如果是最后一次重新启动，最终报告将包含错误消息。
`PermanentFailure`	任务永久失败。如果您请求了任务完成报告，任务将被标记为 `Failed` 并包含错误消息字符串。忽略失败任务的结果字符串。

创建与 S3 批量操作一起使用的 Lambda 函数

此部分提供使用 Lambda 函数必须具有的 AWS Identity and Access Management (IAM) 权限示例。它还包含一个与 S3 批量操作一起使用的示例 Lambda 函数。如果您之前从未创建过 Lambda 函数，请参阅《AWS Lambda 开发人员指南》中的教程：将 AWS Lambda 与 Amazon S3 结合使用。

您必须创建专门与 S3 批量操作一起使用的 Lambda 函数。您无法重复使用基于事件的现有 Amazon S3 Lambda 函数。这是因为用于 S3 批量操作的 Lambda 函数必须接受并返回特殊数据字段。

> **Important**
>
> 用 Java 编写的 AWS Lambda 函数接受 RequestHandler 或 RequestStreamHandler 处理程序接口。但是，为了支持 S3 批量操作请求和响应格式，AWS Lambda 需要 `RequestStreamHandler` 接口来对请求和响应进行自定义序列化和反序列化。此接口允许 Lambda 将 InputStream 和 OutputStream 传递给 Java `handleRequest` 方法。
> 将 Lambda 函数与 S3 批量操作结合使用时，请务必使用 `RequestStreamHandler` 接口。如果您使用 `RequestHandler` 接口，则批处理任务会失败，并在完成报告中显示“Lambda 负载中返回无效的 JSON”。
> 有关更多信息，请参阅《AWS Lambda 用户指南》中的处理程序接口。

IAM 权限示例

以下是将 Lambda 函数与 S3 批量操作结合使用所需的 IAM 权限示例。

Example – S3 批量操作信任策略

以下是您可用于批量操作 IAM 角色的信任策略示例。当您创建任务并授予批量操作代入 IAM 角色的权限时，会指定此 IAM 角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "batchoperations.s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Example – Lambda IAM 策略

以下是提供 S3 批量操作权限以调用 Lambda 函数和读取输入清单的 IAM 策略示例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "BatchOperationsLambdaPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "lambda:InvokeFunction"
            ],
            "Resource": "*"
        }
    ]
}
```

示例请求和响应

本节内容提供 Lambda 函数的请求和响应示例。

Example 请求

以下是 Lambda 函数的请求的 JSON 示例。

```
{
    "invocationSchemaVersion": "1.0",
    "invocationId": "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
    "job": {
        "id": "f3cc4f60-61f6-4a2b-8a21-d07600c373ce"
    },
    "tasks": [
        {
            "taskId": "dGFza2lkZ29lc2hlcmUK",
            "s3Key": "customerImage1.jpg",
            "s3VersionId": "1",
            "s3BucketArn": "arn:aws:s3:us-east-1:0123456788:awsexamplebucket1"
        }
    ]
}
```

Example 响应

以下是 Lambda 函数的响应的 JSON 示例。

```
{
  "invocationSchemaVersion": "1.0",
  "treatMissingKeysAs" : "PermanentFailure",
  "invocationId" : "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
  "results": [
    {
      "taskId": "dGFza2lkZ29lc2hlcmUK",
      "resultCode": "Succeeded",
      "resultString": "[\"Mary Major", \"John Stiles\"]"
    }
  ]
}
```

S3 批量操作的 Lambda 函数示例

以下示例中，Python Lambda 从受版本控制的对象中移除删除标记。

如示例所示，S3 批量操作中的键采用 URL 编码。要将 Amazon S3 与其他 AWS 服务结合使用，请务必对从 S3 批量操作传递的键进行 URL 解码。

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
 obj_key
        )
```

```
        try:
            # If this call does not raise an error, the object version is not a delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
```

```
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

创建调用 Lambda 函数的 S3 批量操作任务

创建 S3 批量操作任务以调用 Lambda 函数时，必须提供以下信息：

- Lambda 函数的 ARN（可能包含函数别名或特定版本号）
- 具有调用此函数的权限的 IAM 角色
- 操作参数 `LambdaInvokeFunction`

有关创建 S3 批量操作任务的更多信息，请参阅 创建 S3 批量操作任务 和 S3 分批操作支持的操作。

以下示例创建使用 AWS CLI 调用 Lambda 函数的 S3 批量操作任务。

```
aws s3control create-job
```

```
    --account-id <AccountID>
    --operation  '{"LambdaInvoke": { "FunctionArn":
 "arn:aws:lambda:Region:AccountID:function:LambdaFunctionName" } }'
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":
{"ObjectArn":"arn:aws:s3:::ManifestLocation","ETag":"ManifestETag"}}'
    --report
 '{"Bucket":"arn:aws:s3:::awsexamplebucket1","Format":"Report_CSV_20180820","Enabled":true,"Pre
    --priority 2
    --role-arn arn:aws:iam::AccountID:role/BatchOperationsRole
    --region Region
    --description “Lambda Function"
```

在 Lambda 清单中提供任务级信息

当您将 AWS Lambda 函数与 S3 批量操作结合使用时，对于所操作的各个任务/键可能会需要附带额外的数据。例如，您可能希望同时提供源对象键和新的对象键。然后，您的 Lambda 函数可以将源键以新名称复制到新 S3 存储桶。默认情况下，通过 Amazon S3 批量操作，您可以仅在任务的输入清单中，指定目标存储桶以及源键列表。下文介绍了如何在清单中包含额外的数据，以便运行更复杂的 Lambda 函数。

要在 S3 批量操作清单中指定各个键的参数以在 Lambda 函数代码中使用，请使用以下 URL 编码的 JSON 格式。key 字段采用类似于 Amazon S3 对象键的方式传递到您的 Lambda 函数。但是，Lambda 函数可以解释它以包含其他值或多个键，如下所示。

> **Note**
>
> 清单中 key 字段的最大字符数为 1024。

Example – 清单使用 JSON 字符串替换“Amazon S3 键”

必须向 S3 批量操作提供 URL 编码的版本。

```
my-bucket,{"origKey": "object1key", "newKey": "newObject1Key"}
my-bucket,{"origKey": "object2key", "newKey": "newObject2Key"}
my-bucket,{"origKey": "object3key", "newKey": "newObject3Key"}
```

Example – URL 编码的清单

必须向 S3 批量操作提供此 URL 编码的版本。非 URL 编码的版本不起作用。

```
my-bucket,%7B%22origKey%22%3A%20%22object1key%22%2C%20%22newKey%22%3A
%20%22newObject1Key%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object2key%22%2C%20%22newKey%22%3A
%20%22newObject2Key%22%7D
my-bucket,%7B%22origKey%22%3A%20%22object3key%22%2C%20%22newKey%22%3A
%20%22newObject3Key%22%7D
```

Example – 具有清单格式的 Lambda 函数将结果写入任务报告

这个 URL 编码的清单示例包含以竖线分隔的对象键，供以下 Lambda 函数进行解析。

```
my-bucket,object1key%7Clower
my-bucket,object2key%7Cupper
my-bucket,object3key%7Creverse
my-bucket,object4key%7Cdelete
```

此 Lambda 函数显示了如何解析编码到 S3 批量操作清单中以竖线分隔的任务。该任务指示应用于指定对象的修订操作。

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.resource("s3")


def lambda_handler(event, context):
    """
    Applies the specified revision to the specified object.

    :param event: The Amazon S3 batch event that contains the ID of the object to
                  revise and the revision type to apply.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
```

```
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]
    # The revision type is packed with the object key as a pipe-delimited string.
    obj_key, revision = parse.unquote(task["s3Key"], encoding="utf-8").split("|")
    bucket_name = task["s3BucketArn"].split(":")[-1]

    logger.info("Got task: apply revision %s to %s.", revision, obj_key)

    try:
        stanza_obj = s3.Bucket(bucket_name).Object(obj_key)
        stanza = stanza_obj.get()["Body"].read().decode("utf-8")
        if revision == "lower":
            stanza = stanza.lower()
        elif revision == "upper":
            stanza = stanza.upper()
        elif revision == "reverse":
            stanza = stanza[::-1]
        elif revision == "delete":
            pass
        else:
            raise TypeError(f"Can't handle revision type '{revision}'.")

        if revision == "delete":
            stanza_obj.delete()
            result_string = f"Deleted stanza {stanza_obj.key}."
        else:
            stanza_obj.put(Body=bytes(stanza, "utf-8"))
            result_string = (
                f"Applied revision type '{revision}' to " f"stanza {stanza_obj.key}."
            )

        logger.info(result_string)
        result_code = "Succeeded"
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            result_code = "Succeeded"
            result_string = (
                f"Stanza {obj_key} not found, assuming it was deleted "
```

```
                f"in an earlier revision."
            )
            logger.info(result_string)
        else:
            result_code = "PermanentFailure"
            result_string = (
                f"Got exception when applying revision type '{revision}' "
                f"to {obj_key}: {error}."
            )
            logger.exception(result_string)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

从 S3 批量操作教程中学习

以下教程提供了使用 Lambda 的一些批量操作任务的完整端到端过程。

- 教程：使用 S3 批量操作、AWS Lambda 和 AWS Elemental MediaConvert 对视频进行批量转码

替换所有对象标签

替换所有对象标签操作将替换清单中列出的每个对象上的 Amazon S3 对象标签。Amazon S3 对象标签是一个字符串的键/值对，您可用它来存储有关对象的元数据。

要创建替换所有对象标签任务，您需要提供要应用的标签集。S3 分批操作将相同的标签集应用于每个对象。您提供的标签集替换已与清单中的对象关联的任何标签集。S3 分批操作不支持在保留现有标签的同时向对象添加标签。

如果清单中的对象位于受版本控制的存储桶中，您可将标签集应用到每个对象的特定版本。您通过为清单中的每个对象指定版本 ID 来完成此操作。如果您没有包括任何对象的版本 ID，S3 分批操作将标签集应用到每个对象的最新版本。

限制和局限性

- 您指定用于运行分批操作任务的 AWS Identity and Access Management (IAM) 角色必须有权执行基础 Amazon S3 替换所有对象标签操作。有关所需权限的更多信息，请参阅 Amazon Simple Storage Service API 参考中的 PutObjectTagging。
- S3 分批操作使用 Amazon S3 PutObjectTagging 操作，将标签应用到清单中的每个对象。适用于基础操作的所有限制也适用于 S3 分批操作任务。

有关使用控制台创建任务的更多信息，请参阅创建 S3 分批操作任务。

有关对象标签的更多信息，请参阅本指南中的 使用标签对存储进行分类 以及 Amazon Simple Storage Service API 参考中的 PutObjectTagging、GetObjectTagging 和 DeleteObjectTagging。

删除所有对象标签

删除所有对象标签操作将删除当前与清单中列出的对象关联的所有 Amazon S3 对象标签集。S3 分批操作不支持在保留其他标签的同时删除对象中的标签。

如果清单中的对象位于受版本控制的存储桶中，则可以从对象的特定版本中删除标签集。通过为清单中的每个对象指定版本 ID 来完成此操作。如果您没有包括对象的版本 ID，S3 分批操作将从每个对象的最新版本中删除该标签集。

有关分批操作清单的更多信息，请参阅 指定清单。

> **Warning**
>
> 运行此任务将删除清单中列出的每个对象上的所有对象标签集。

限制和局限性

- 您指定用于运行任务的 AWS Identity and Access Management (IAM) 角色必须有权执行基础 Amazon S3 Delete 对象标签操作。有关更多信息，请参阅 Amazon Simple Storage Service API 参考中的 DeleteObjectTagging。
- S3 分批操作使用 Amazon S3 DeleteObjectTagging 操作从清单中的每个对象删除标签集。适用于基础操作的所有限制也适用于 S3 分批操作任务。

有关创建任务的信息，请参阅 创建 S3 批量操作任务。

有关对象标签的更多详细信息，请参阅本指南中的 替换所有对象标签 以及 Amazon Simple Storage Service API 参考中的 PutObjectTagging、GetObjectTagging 和 DeleteObjectTagging。

替换访问控制列表

替换访问控制列表 (ACL) 操作替换在清单中列出的每个对象的 Amazon S3 访问控制列表 (ACL)。使用 ACL，您可以定义谁可以访问某个对象，以及他们可以执行哪些操作。

S3 分批操作支持您定义的自定义 ACL 以及 Amazon S3 随预定义的访问权限集提供的标准 ACL。

如果清单中的对象位于受版本控制的存储桶中，您可将 ACL 应用到每个对象的特定版本。您通过为清单中的每个对象指定版本 ID 来完成此操作。如果您没有包括任何对象的版本 ID，S3 分批操作将 ACL 应用到对象的最新版本。

有关 Amazon S3 中 ACL 的更多信息，请参阅 访问控制列表 (ACL) 概述。

S3 阻止公有访问

如果您希望限制对存储桶中所有对象的公有访问，则应使用 Amazon S3 阻止公有访问而不是 S3 分批操作。阻止公有访问可以在各个存储桶或整个账户的范围进行，并且只需一个简单的操作，立即生效。当您的目标是控制某个存储桶或账户中所有对象的公有访问时，这是一个更好的选择。在您需要将自定义 ACL 应用到清单中的每个对象时，请使用 S3 分批操作。有关 S3 阻止公有访问的更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

S3 对象所有权

如果清单中的对象位于桶中，而该桶使用桶拥有者强制执行的对象所有权设置，则 Replace access control list (ACL) [替换访问控制列表（ACL）] 操作只能指定向桶拥有者授予完全控制权限的对象 ACL。该操作无法将对象 ACL 权限授予其他 AWS 账户 或组。有关更多信息，请参阅为您的存储桶控制对象所有权和禁用 ACL。。

限制和局限性

- 您指定用于运行替换访问控制列表任务的角色必须有权执行基础 Amazon S3 `PutObjectAcl` 操作。有关所需权限的更多信息，请参阅 Amazon Simple Storage Service API 参考中的 PutObjectAcl。
- S3 分批操作使用 Amazon S3 `PutObjectAcl` 操作，将指定的 ACL 应用到清单中的每个对象。因此，适用于基础 PutObjectAcl 操作的所有限制也适用于 S3 分批操作 Replace 访问控制列表任务。

使用批量操作还原对象

还原操作启动对清单中列出的已归档 Amazon S3 对象的还原请求。必须先还原以下归档对象，然后才能实时访问这些对象：

- 在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中归档的对象
- 通过归档访问层或深度归档访问层中的 S3 Intelligent-Tiering 存储类归档的对象

在 S3 分批操作任务中使用 S3 启动还原对象操作会为清单中指定的每个对象发出还原请求。

> **Important**
>
> S3 启动还原对象任务仅启动还原对象的请求。在为每个对象启动请求后，S3 分批操作将对象的任务报告为完成。还原对象时，Amazon S3 不会更新任务或以其他方式通知您。不过，当对象在 Amazon S3 中可用时，您可以使用 S3 事件通知来接收通知。有关更多信息，请参阅 Amazon S3 事件通知。

当您创建 S3 启动还原对象任务时，可以使用以下参数：

ExpirationInDays

此参数用于指定 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 对象在 Amazon S3 中保持可用的时长。启动以 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 对象为目标的还原对象任务要求您将 `ExpirationInDays` 设置为 1 或更大的值。

> **Important**
>
> 在创建以 S3 Intelligent-Tiering 归档访问和深度归档访问层对象为目标的 S3 启动还原对象操作任务时，请勿设置 `ExpirationInDays`。S3 Intelligent-Tiering 归档访问层中的对象不受还原过期的限制，因此指定 `ExpirationInDays` 会导致还原请求失败。

GlacierJobTier

Amazon S3 可以使用三个不同的检索层之一还原对象：`EXPEDITED`、`STANDARD` 和 `BULK`。但是，S3 批量操作功能仅支持 `STANDARD` 检索层。有关各检索层之间的差异的更多信息，请参阅归档检索选项。

有关每层定价的更多信息，请参阅 Amazon S3 定价页面上的请求和数据检索部分。

从 S3 Glacier 和 S3 Intelligent-Tiering 还原时的区别

从 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类还原归档文件与从归档访问层或深度归档访问层中的 S3 Intelligent-Tiering 存储类恢复文件不同。

- 当您从 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 中还原时，将创建对象的临时副本。Amazon S3 会在 `ExpirationInDays` 参数中指定的值过期后删除此副本。删除此临时副本后，您必须提交额外的还原请求才能访问该对象。
- 还原归档的 S3 Intelligent-Tiering 对象时，请勿指定 `ExpirationInDays` 参数。当您从 S3 Intelligent-Tiering 归档访问层或深度归档访问层还原对象时，此对象会转换回 S3 Intelligent-Tiering 频繁访问层中。在至少连续 90 天无访问后，对象会自动转换到归档访问层。在至少连续 180 天无访问后，对象会自动移动到深度归档访问层。
- 批量操作任务既可以在 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类对象上运行，也可以在 S3 Intelligent-Tiering 归档访问和深度归档访问存储层对象上运行。批量操作不能在同一个任务中对两种类型的归档对象进行操作。要还原两种类型的对象，必须创建单独的分批操作任务。

重叠还原

如果您的 S3 启动还原对象任务尝试还原已处于还原过程中的对象，S3 分批操作的操作过程如下所示：

如果满足以下任意条件，对象的还原操作将成功：

- 与已在进行的还原请求相比，此任务的 ExpirationInDays 值相同，而其 GlacierJobTier 值更快。
- 之前的还原请求已完成，对象当前已可用。在这种情况下，批量操作会更新已还原对象的过期日期，以便与在正在进行的还原请求中指定的 ExpirationInDays 值匹配。

如果满足以下任意条件，对象的还原操作失败：

- 已在进行的还原操作尚未完成，并且此任务的还原持续时间（由 ExpirationInDays 值指定）不同于在正在进行的还原请求中指定的还原持续时间。
- 此任务的还原层（由 GlacierJobTier 值指定）与在正进行的还原请求中指定的还原层速度相同或更慢。

限制

S3 启动还原对象任务有以下限制：

- 您必须在与归档对象相同的区域中创建任务。
- S3 分批操作不支持 EXPEDITED 检索层。

有关还原对象的更多信息，请参阅 恢复已归档的对象。

S3 对象锁定保留

对象锁定保留操作允许您使用治理模式或合规性模式为对象应用保留日期。这些保留模式会应用不同级别的保护。可将任一保留模式应用于任意对象版本。与依法保留类似，保留日期可防止覆盖或删除对象。Amazon S3 存储在对象元数据中指定的保留终止日期，并保护指定版本的对象版本，直到保留期限到期。

您可以将 S3 分批操作与对象锁定一起使用，以便同时管理许多 Amazon S3 对象的保留日期。您可以在清单中指定目标对象的列表并将该列表提交到分批操作以完成操作。有关更多信息，请参阅 S3 对象锁定the section called “保留期限”。

具有保留日期的任务 S3 分批操作将持续运行，直至达到完成、取消或失败状态。在要通过单个请求添加、更改或删除多个对象的保留日期时，应使用 S3 分批操作和 S3 对象锁定保留。

在处理清单中的任何密钥之前，分批操作会验证是否已在您的存储桶上启用对象锁定。要执行操作和验证，分批操作需要 IAM 角色中的 s3:GetBucketObjectLockConfiguration 和

`s3:PutObjectRetention` 权限，以允许分批操作代表您调用对象锁定。有关更多信息，请参阅 the section called “对象锁定注意事项”。

有关将此操作与 REST API 结合使用的信息，请参阅《Amazon Simple Storage Service API 参考》的 CreateJob 操作中的 `S3PutObjectRetention`。

有关使用此操作的 AWS Command Line Interface示例，请参阅the section called “使用分批操作与对象锁定保留”。有关 AWS SDK for Java 示例，请参阅the section called “使用分批操作与对象锁定保留”。

限制和局限性

- S3 分批操作不会进行任何存储桶级别的更改。
- 必须在执行任务的存储桶上配置版本控制和 S3 对象锁定。
- 清单中列出的所有对象都必须位于同一个存储桶中。
- 除非清单中明确指定了版本，否则该操作将适用于最新版本的对象。
- 您需要 IAM 角色中的 `s3:PutObjectRetention` 权限才能使用此操作。
- `s3:GetBucketObjectLockConfiguration`需要 IAM 权限才能确认是否已为 S3 存储桶启用对象锁定。
- 您只能在应用 `COMPLIANCE` 模式保留日期的情况下延长对象的保留期限，并且不能将其缩短。

S3 对象锁定依法保留

对象锁定依法保留操作使您能够对对象版本实施依法保留。与设置保留期限相似，依法保留可防止对象版本被覆盖或删除。但是，依法保留没有关联的保留期限，在删除之前将一直有效。

您可以将 S3 分批操作与对象锁定一起使用，以便同时添加对很多 Amazon S3 对象的依法保留。可以通过在清单中列出目标对象并将该列表提交到分批操作来执行此操作。具有对象锁定依法保留的 S3 分批操作任务将持续运行，直至达到完成、取消或失败状态。

在处理清单中的任何键之前，S3 分批操作会验证是否已在您的 S3 存储桶上启用对象锁定。要执行对象操作和存储桶级别验证，S3 分批操作需要 IAM 角色中的 `s3:PutObjectLegalHold` 和 `s3:GetBucketObjectLockConfiguration`，以允许 S3 分批操作代表您调用 S3 对象锁定。

当您创建 S3 分批操作任务以删除依法保留时，只需将依法保留状态指定为禁用。有关更多信息，请参阅 the section called “对象锁定注意事项”。

有关如何将此操作与 REST API 结合使用的信息，请参阅 Amazon Simple Storage Service API 参考的 CreateJob 操作中的 `S3PutObjectLegalHold`。

有关此操作的示例用法，请参阅 使用适用于 Java 的 AWS 开发工具包。

限制和局限性

- S3 分批操作不会进行任何存储桶级别的更改。
- 清单中列出的所有对象都必须位于同一个存储桶中。
- 必须在执行任务的存储桶上配置版本控制和 S3 对象锁定。
- 除非清单中明确指定了版本，否则该操作将适用于最新版本的对象。
- `s3:PutObjectLegalHold`IAM 角色需要 权限才能在对象中添加或删除依法保留。
- `s3:GetBucketObjectLockConfiguration`需要 IAM 权限才能确认是否已为 S3 存储桶启用 S3 对象锁定。

- 复制对象
- 调用 AWS Lambda 函数
- 替换所有对象标签
- 删除所有对象标签
- 替换访问控制列表
- 使用批量操作还原对象
- S3 对象锁定保留
- S3 对象锁定依法保留
- 使用 S3 分批复制以复制现有对象

管理 S3 分批操作任务

Amazon S3 提供一组功能强大的工具，帮助您在创建 S3 分批操作任务后管理任务。本节介绍了可以使用 AWS Management Console、AWS CLI、AWS 开发工具包或 REST API 管理和跟踪任务的操作。

主题

- 使用 Amazon S3 控制台管理 S3 分批操作任务
- 列出任务
- 查看任务详细信息

- 分配任务优先级

使用 Amazon S3 控制台管理 S3 分批操作任务

使用控制台，您可以管理 S3 分批操作任务。例如，您可以：

- 查看活动任务和已排队的任务
- 更改工作的优先级
- 确认并运行任务
- 克隆任务
- 取消任务

要使用控制台管理分批操作

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，请选择分批操作。
3. 请选择您想要管理的特定任务。

列出任务

您可以检索 S3 分批操作任务的列表。此列表包含您尚未完成的任务，以及您在过去 90 天内完成的任务。该任务列表包含每个任务的信息，如其 ID、描述、优先级、当前状态以及成功和失败的任务数。您可以按状态筛选任务列表。当您通过控制台检索任务列表时，还可以按描述或 ID 搜索任务并按 AWS 区域 对其进行筛选。

获取“Active (活动)”任务和“Complete (完成)”任务的列表

以下 AWS CLI 示例获取 `Active` 和 `Complete` 任务的列表。

```
aws s3control list-jobs \
    --region us-west-2 \
    --account-id acct-id \
    --job-statuses '["Active","Complete"]' \
    --max-results 20
```

有关更多信息和示例，请参阅 AWS CLI 命令参考中的 list-jobs。

查看任务详细信息

如果您需要的信息比通过列出任务来检索的任务信息多，则可查看单个任务的所有详细信息。您可以查看您尚未完成的作业或者过去 90 天内完成的作业的详细信息。除了任务列表中返回的信息之外，单个任务的详细信息还包含其他项目，例如：

- 操作参数
- 有关清单的详细信息
- 有关完成报告的信息（如果您在创建任务时配置了完成报告）
- 您分配用于运行任务的用户角色的 Amazon Resource Name (ARN)

通过查看单个任务的详细信息，可以访问任务的整个配置。要查看任务的详细信息，您可以使用 Amazon S3 控制台或 AWS Command Line Interface（AWS CLI）。

在 Amazon S3 控制台中获取 S3 批量操作任务的描述

使用控制台查看批量操作任务的描述

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择批量操作。
3. 选择特定任务的任务 ID，以查看其详细信息。

在 AWS CLI 中获取 S3 批量操作任务的描述

以下示例使用 AWS CLI 获取 S3 批量操作任务的描述。要使用以下示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control describe-job \
--region us-west-2 \
--account-id acct-id \
--job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

有关更多信息和示例，请参阅 AWS CLI 命令参考中的 describe-job。

分配任务优先级

您可以为每个任务分配一个数字优先级，该优先级可以是任意正整数。S3 分批操作根据分配的优先级确定任务的优先次序。系统将首先评估具有较高优先级（或者优先级参数的数值较高）的任务。优先级按降序确定。例如，优先级值为 10 的任务队列将会比优先级值为 1 的任务队列优先计划。

您可以在任务正在运行时更改其优先级。如果您在任务运行时提交优先级更高的新任务，则优先级较低的任务可以暂停，以允许优先级高的任务运行。

更改任务优先级不会影响任务处理速度。

> **Note**
>
> S3 分批操作尽最大努力遵循任务优先级。尽管优先级高的任务通常优先于优先级低的任务，但 Amazon S3 无法保证任务的严格顺序。

使用 S3 控制台

如何在 AWS Management Console 中更新任务优先级

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，请选择分批操作。
3. 请选择您想要管理的特定任务。
4. 请选择 Action（操作）。请在下拉列表中，选择 Update priority（更新优先级）。

使用 AWS CLI

以下示例使用 AWS CLI 更新任务优先级。数字越大，执行优先级越高。

```
aws s3control update-job-priority \
    --region us-west-2 \
    --account-id acct-id \
    --priority 98 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

使用 AWS SDK for Java

以下示例使用 AWS SDK for Java 更新 S3 分批操作任务的优先级。

有关任务优先级的更多信息，请参阅 分配任务优先级。

Example

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobPriorityRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobPriority {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobPriority(new UpdateJobPriorityRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withPriority(98));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

跟踪任务状态和完成报告

借助 S3 分批操作，您可以查看和更新任务状态、添加通知和日志记录、跟踪任务失败以及生成完成报告。

主题

任务状态

在您创建和运行任务之后，该任务会逐步经过一系列状态。下表描述了状态以及状态之间可能的转换。

状态	说明	转换
New	在您创建任务时，任务首先处于 New 状态。	Amazon S3 开始处理清单对象时，任务自动转入 Preparing 状态。
Preparing	Amazon S3 正在处理清单对象及其他任务参数，以设置和运行任务。	Amazon S3 完成清单和其他参数的处理之后，任务自动转入 Ready 状态。然后，它随时可以在清单中列出的对象上开始运行指定的操作。 如果任务要求在运行之前确认，就像您在使用 Amazon

状态	说明	转换
		S3 控制台创建任务时那样，则任务从 Preparing 转换为 Suspended 。在您确认希望运行之前，任务将保持在 Suspended 状态。
Suspended	任务需要确认，但您尚未确认希望运行任务。只有使用 Amazon S3 控制台创建的任务需要确认。使用控制台创建的任务在 Suspended 之后立即进入 Preparing 状态。在您确认希望运行任务并且任务成为 Ready 状态之后，它不会再返回 Suspended 状态。	在您确认希望运行任务之后，其状态将更改为 Ready。
Ready	Amazon S3 已准备好开始运行请求的对象操作。	Amazon S3 开始运行任务时，任务自动转入 Active 状态。任务保持在 Ready 状态的时间长度取决于您是否已经在运行具有较高优先级的任务，以及完成这些任务需要多长时间。
Active	Amazon S3 正在对清单中列出的对象执行请求的操作。当任务处于 Active 状态时，您可使用 Amazon S3 控制台，或者通过 REST API、AWS CLI 或 AWS 开发工具包使用 DescribeJob 操作监控其进度。	对象上没有运行的操作时，任务移出 Active 状态。此过程自动进行，例如在任务成功完成或失败时。或者，此过程也可作为用户操作的结果发生，例如取消某个任务。任务转入的状态取决于发生转换的原因。
Pausing	任务从其他状态转换为 Paused。	Paused 阶段完成时，任务自动转入 Pausing 状态。

状态	说明	转换
Paused	在当前任务正在运行时，如果您提交了具有更高优先级的其他任务，任务可以进入 Paused 状态。	对于处于 Paused 状态的任务，如果阻止了该任务的任何较高优先级的任务执行完成、失败或暂停，则该任务自动返回 Active 状态。
Complete	任务已完成对清单中所有对象执行请求的操作。对于每个对象，操作可能成功或失败。如果您配置任务生成完成报告，则在任务处于 Complete 状态之后报告立即可用。	Complete 是最终状态。任务进入 Complete 状态之后，将不会转换为任何其他状态。
Cancelling	任务转换为 Cancelled 状态。	Cancelled 阶段完成时，任务自动转入 Cancelling 状态。
Cancelled	您请求取消任务，S3 分批操作已成功取消任务。任务不会提交任何新请求到 Amazon S3。	Cancelled 是最终状态。任务进入 Cancelled 状态之后，它不会转换为任何其他状态。
Failing	任务转换为 Failed 状态。	Failed 阶段完成后，任务自动转入 Failing 状态。
Failed	任务已失败，不再运行。有关任务失败的更多信息，请参阅跟踪任务失败。	Failed 是最终状态。任务进入 Failed 状态之后，它不会转换为任何其他状态。

更新任务状态

以下 AWS CLI 和适用于 Java 的开发工具包示例更新了分批操作任务的状态。有关使用 S3 控制台管理分批操作任务的更多信息，请参阅 使用 Amazon S3 控制台管理 S3 分批操作任务。

使用 AWS CLI

- 如果您在上一个 `--no-confirmation-required` 示例中未指定 `create-job` 参数，则任务将保持暂停状态，直至您通过将其状态设置为 `Ready` 来确认任务。然后，Amazon S3 会使任务符合执行的条件。

```
aws s3control update-job-status \
    --region us-west-2 \
    --account-id 181572960644 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
    --requested-job-status 'Ready'
```

- 通过将任务状态设置为 `Cancelled` 来取消任务。

```
aws s3control update-job-status \
     --region us-west-2 \
     --account-id 181572960644 \
     --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
     --status-update-reason "No longer needed" \
     --requested-job-status Cancelled
```

使用适用于 Java 的 AWS 开发工具包

以下示例使用 AWS SDK for Java 更新 S3 分批操作任务的状态。

有关任务状态的更多信息，请参阅 跟踪任务状态和完成报告。

Example

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.UpdateJobStatusRequest;
```

```
import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateJobStatus {
    public static void main(String[] args) {
        String accountId = "Account ID";
        String jobId = "00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.updateJobStatus(new UpdateJobStatusRequest()
                    .withAccountId(accountId)
                    .withJobId(jobId)
                    .withRequestedJobStatus("Ready"));

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

通知和日志记录

除了请求完成报告外，您还可以使用 AWS CloudTrail 捕获、查看和审核分批操作活动 由于分批操作使用现有的 Amazon S3 API 来执行任务，所以这些任务还会发出与您直接调用这些 API 时相同的事件。因此，您可以使用相同的通知、日志记录和审核工具以及已用于 Amazon S3 的进程来跟踪和记录您的任务及其所有任务的进度。有关更多信息，请参阅以下部分中的示例。

> **Note**
>
> Amazon S3 分批操作在任务执行期间在 CloudTrail 中生成管理事件和数据事件。这些事件的数量随每个任务清单中的键数量而扩展。有关详细信息，请参阅 CloudTrail 定价页面，其中包

括定价如何根据您在账户中配置的跟踪数量而变化的示例。要了解如何配置和记录事件以满足您的需求，请参阅《AWS CloudTrail 用户指南》中的创建您的第一个跟踪。

有关 Amazon S3 事件的更多信息，请参阅 Amazon S3 事件通知。

跟踪任务失败

如果 S3 分批操作任务遇到的问题使其无法成功运行（如无法读取指定清单），则该任务失败。任务失败时，系统会生成一个或多个故障代码或失败原因。S3 分批操作将故障代码和原因与任务一起存储，以便于您通过请求任务的详细信息来查看。如果您请求任务的完成报告，同时还会显示故障代码和原因。

为防止任务运行大量不成功的操作，Amazon S3 会对每个分批操作任务施加任务失败阈值。当任务已运行至少 1000 个任务时，Amazon S3 将监控任务失败率。在任何时刻，如果失败率（失败的任务数占已运行任务总数的比例）超过 50%，则任务失败。如果任务由于超过了任务失败阈值而失败，您可以确定造成失败的原因。例如，您可能在清单中意外地包括了一些指定存储桶中不存在的对象。修复错误之后，您可以重新提交任务。

> **Note**
>
> S3 分批操作异步操作，任务不一定按照清单中所列的对象顺序来运行。因此您不能使用清单顺序来确定哪些对象的任务成功，哪些对象的任务失败。而是可以通过查看任务的完成报告（如果已请求）或查看 AWS CloudTrail 事件日志来帮助确定失败原因。

完成报告

在创建任务时，您可以请求完成报告。只要 S3 分批操作成功调用至少一个任务，Amazon S3 便会在运行任务完成、失败或被取消后生成完成报告。可对完成报告进行配置，以包含所有任务或只包含失败的任务。

完成报告包含任务配置以及每个任务的状态和信息，包括对象键和版本、状态、错误代码以及任何错误的描述。完成报告提供了以合并格式查看任务结果的简单途径，且无需进行任何附加设置。完成报告使用 Amazon S3 托管式密钥（SSE-S3）进行加密。有关完成报告的示例，请参阅 示例：S3 分批操作完成报告。

如果您未配置完成报告，则仍可使用 CloudTrail 和 Amazon CloudWatch 监控并审核您的任务及其任务。有关更多信息，请参阅以下部分。

主题

示例：通过 AWS CloudTrail 在 Amazon EventBridge 中跟踪 S3 分批操作任务。

Amazon S3 分批操作任务活动在 AWS CloudTrail 中记录为事件。您可以在 Amazon EventBridge 中创建自定义规则，然后将这些事件发送到您选择的目标通知资源，例如 Amazon Simple Notification Service (Amazon SNS)。

> **Note**
>
> Amazon EventBridge 是管理事件的首选方式。Amazon CloudWatch Events 和 EventBridge 是相同的底层服务和 API，但 EventBridge 提供了更多功能。您在 CloudWatch 或 EventBridge 中所作的更改将显示在每个控制台中。有关更多信息，请参阅 Amazon EventBridge 用户指南。

跟踪示例

在 CloudTrail 中记录的 S3 分批操作事件

创建分批操作任务后，它将作为 `JobCreated` 事件记录在 CloudTrail 中。在任务运行时，它会在处理过程中更改状态，并且其他 `JobStatusChanged` 事件将记录到 CloudTrail 中。您可以在 CloudTrail 控制台中查看这些事件。有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

> **Note**
>
> CloudTrail 中只记录 S3 分批操作任务 `status-change` 事件。

Example CloudTrail 记录的 S3 分批操作任务完成事件

```
{
    "eventVersion": "1.05",
```

```
    "userIdentity": {
        "accountId": "123456789012",
        "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-05T18:25:30Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "f907577b-bf3d-4c53-b9ed-8a83a118a554",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "123412341234",
    "serviceEventDetails": {
        "jobId": "d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "jobArn": "arn:aws:s3:us-west-2:181572960644:job/
d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "status": "Complete",
        "jobEventId": "b268784cf0a66749f1a05bce259804f5",
        "failureCodes": [],
        "statusChangeReason": []
    }
}
```

用于跟踪 S3 分批操作任务事件的 EventBridge 规则

以下示例说明如何在 Amazon EventBridge 中创建规则，以便将 AWS CloudTrail 记录的 S3 分批操作事件捕获到您选择的目标。

为此，您可以按照创建用于响应事件的 EventBridge 规则中的所有步骤来创建规则。粘贴以下 S3 分批操作自定义事件模式策略（如果适用），然后选择所需的目标服务。

S3 批处理操作自定义事件模式策略

```
{
    "source": [
        "aws.s3"
    ],
    "detail-type": [
        "AWS Service Event via CloudTrail"
```

```
    ],
    "detail": {
        "eventSource": [
            "s3.amazonaws.com"
        ],
        "eventName": [
            "JobCreated",
            "JobStatusChanged"
        ]
    }
}
```

以下示例是从 EventBridge 事件规则发送到 Amazon Simple Queue Service (Amazon SQS) 的两个分批操作事件。由于分批操作任务在处理期间会经历多种不同的状态（New、Preparing、Active 等），因此对于每个任务，您都将收到多条消息。

Example JobCreated 示例事件

```
{
    "version": "0",
    "id": "51dc8145-541c-5518-2349-56d7dffdf2d8",
    "detail-type": "AWS Service Event via CloudTrail",
    "source": "aws.s3",
    "account": "123456789012",
    "time": "2020-02-27T15:25:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventVersion": "1.05",
        "userIdentity": {
            "accountId": "11112223334444",
            "invokedBy": "s3.amazonaws.com"
        },
        "eventTime": "2020-02-27T15:25:49Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "JobCreated",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "s3.amazonaws.com",
        "userAgent": "s3.amazonaws.com",
        "eventID": "7c38220f-f80b-4239-8b78-2ed867b7d3fa",
        "readOnly": false,
        "eventType": "AwsServiceEvent",
```

```
        "serviceEventDetails": {
            "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
            "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
            "status": "New",
            "jobEventId": "f177ff24f1f097b69768e327038f30ac",
            "failureCodes": [],
            "statusChangeReason": []
        }
    }
}
```

Example JobStatusChanged 任务完成事件

```
{
  "version": "0",
  "id": "c8791abf-2af8-c754-0435-fd869ce25233",
  "detail-type": "AWS Service Event via CloudTrail",
  "source": "aws.s3",
  "account": "123456789012",
  "time": "2020-02-27T15:26:42Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "eventVersion": "1.05",
    "userIdentity": {
      "accountId": "1111222233334444",
      "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-27T15:26:42Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "eventID": "0238c1f7-c2b0-440b-8dbd-1ed5e5833afb",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "serviceEventDetails": {
      "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
      "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
      "status": "Complete",
```

```
      "jobEventId": "51f5ac17dba408301d56cd1b2c8d1e9e",
      "failureCodes": [],
      "statusChangeReason": []
    }
  }
}
```

示例：S3 分批操作完成报告

当您创建 S3 分批操作任务时，可以为全部任务或仅失败任务请求完成报告。只要成功调用了至少一个任务，S3 分批操作即会为已完成、已失败或已取消的任务生成报告。

完成报告包含每个任务的其他信息，包括对象键名称和版本、版本、错误代码以及任何错误的描述。每个失败任务的错误的描述可用于诊断在任务创建期间发生的问题，如权限。

> **Note**
>
> 完成报告始终使用 Amazon S3 托管式密钥（SSE-S3）进行加密。

Example 顶级清单结果文件

顶级 `manifest.json` 文件包含每个成功报告的位置以及（如果任务有任何失败）失败报告的位置，如以下示例所示。

```
{
    "Format": "Report_CSV_20180820",
    "ReportCreationDate": "2019-04-05T17:48:39.725Z",
    "Results": [
        {
            "TaskExecutionStatus": "succeeded",
            "Bucket": "my-job-reports",
            "MD5Checksum": "83b1c4cbe93fc893f54053697e10fd6e",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/
results/6217b0fab0de85c408b4be96aeaca9b195a7daa5.csv"
        },
        {
            "TaskExecutionStatus": "failed",
            "Bucket": "my-job-reports",
            "MD5Checksum": "22ee037f3515975f7719699e5c416eaa",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/results/
b2ddad417e94331e9f37b44f1faf8c7ed5873f2e.csv"
```

```
        }
    ],
    "ReportSchema": "Bucket, Key, VersionId, TaskStatus, ErrorCode, HTTPStatusCode,
 ResultMessage"
}
```

Example 失败任务报告

失败的任务报告包含所有失败 任务的以下信息：

- Bucket
- Key
- VersionId
- TaskStatus
- ErrorCode
- HTTPStatusCode
- ResultMessage

以下示例报告显示 AWS Lambda 函数超时，从而导致失败以超出失败阈值的情况。然后，它会标记为 PermanentFailure。

```
awsexamplebucket1,image_14975,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:21.155Z 2845ca0d-38d9-4c4b-
abcf-379dc749c452 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_15897,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:29.610Z 2d0a330b-de9b-425f-
b511-29232fde5fe4 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_14819,,failed,200,PermanentFailure,"Lambda returned function
 error: {""errorMessage"":""2019-04-05T17:35:22.362Z fcf5efde-74d4-4e6d-b37a-
c7f18827f551 Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_15930,,failed,200,PermanentFailure,"Lambda returned function
 error: {""errorMessage"":""2019-04-05T17:35:29.809Z 3dd5b57c-4a4a-48aa-8a35-
cbf027b7957e Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_17644,,failed,200,PermanentFailure,"Lambda
 returned function error: {""errorMessage"":""2019-04-05T17:35:46.025Z
 10a764e4-2b26-4d8c-9056-1e1072b4723f Task timed out after 3.00 seconds""}"
awsexamplebucket1,image_17398,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:44.661Z 1e306352-4c54-4eba-
aee8-4d02f8c0235c Task timed out after 3.00 seconds""}"
```

Example 成功任务报告

成功任务报告包含成功 任务的以下各项内容：

- Bucket
- Key
- VersionId
- TaskStatus
- ErrorCode
- HTTPStatusCode
- ResultMessage

在以下示例中，Lambda 函数将 Amazon S3 对象成功复制到其他存储桶。返回的 Amazon S3 响应会传回到 S3 分批操作，然后写入最终完成报告。

```
awsexamplebucket1,image_17775,,succeeded,200,,"{u'CopySourceVersionId':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'RequestId': '3ED5852152014362', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'x-amz-copy-source-version-id':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', 'server': 'AmazonS3', 'x-amz-request-id':
 '3ED5852152014362', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT', 'content-type':
 'application/xml'}}}"
awsexamplebucket1,image_17763,,succeeded,200,,"{u'CopySourceVersionId':
 '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()),
 u'ETag': '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata':
 {'HTTPStatusCode': 200, 'RetryAttempts': 0, 'HostId': 'GiCZNYr8LHd/
Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'RequestId':
 '1BC9F5B1B95D7000', 'HTTPHeaders': {'content-length': '234', 'x-amz-id-2':
 'GiCZNYr8LHd/Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'x-
amz-copy-source-version-id': '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', 'server': 'AmazonS3',
 'x-amz-request-id': '1BC9F5B1B95D7000', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT',
 'content-type': 'application/xml'}}}"
awsexamplebucket1,image_17860,,succeeded,200,,"{u'CopySourceVersionId':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', u'CopyObjectResult': {u'LastModified':
```

```
 datetime.datetime(2019, 4, 5, 17, 35, 40, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'RequestId': '8D9CA56A56813DF3', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'x-amz-copy-source-version-id':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', 'server': 'AmazonS3', 'x-amz-request-id':
 '8D9CA56A56813DF3', 'date': 'Fri, 05 Apr 2019 17:35:40 GMT', 'content-type':
 'application/xml'}}}"
```

使用标签控制访问和标记任务

您可以通过添加标签来标记和控制对 S3 分批操作任务的访问。标签可用于确定谁负责分批操作任务。任务标签的存在可授予或限制用户取消任务、激活处于确认状态的任务或更改任务优先级的能力。您可以创建附加了标签的任务，并且可以在创建任务后向任务添加标签。每个标签都是一个键值对，可以在创建任务或稍后更新时包含该键值对。

> **Warning**
>
> 任务标签不应包含任何机密信息或个人数据。

考虑以下标记示例：假设您希望您的财务部门创建一个分批操作任务。您可以编写允许用户调用 `CreateJob` 的 AWS Identity and Access Management (IAM) 策略，前提是创建任务时使用分配了 `Finance` 值的 `Department` 标签。此外，您可以将该策略附加到作为财务部门成员的所有用户。

继续使用此示例，您可以编写一个策略，允许用户更新具有所需标签的任何任务的优先级，或者取消具有这些标签的任何任务。有关更多信息，请参阅 the section called “控制权限”。

您可以在创建新的 S3 分批操作任务时向其添加标签，也可以将向现有任务添加标签。

注意以下标签限制：

- 只要任务具有唯一的标签键，就可以将最多 50 个标签与该任务关联。
- 标签键的长度最大可以为 128 个 Unicode 字符，标签值的长度最大可以为 256 个 Unicode 字符。
- 键和值区分大小写。

有关标签限制的更多信息，请参阅《AWS Billing and Cost Management 用户指南》中的用户定义的标签限制。

与 S3 分批操作任务标记相关的 API 操作

Amazon S3 支持以下特定于 S3 分批操作任务标记的 API 操作：

- GetJobTagging – 返回与分批操作任务关联的标签集。
- PutJobTagging – 替换与任务关联的标签集。使用此 API 操作进行 S3 分批操作任务标签管理有两种不同的方案：
  - 任务没有标签 – 您可以向任务添加一组标签（任务之前没有标签）。
  - 任务具有一组现有标签 – 要修改现有标签集，您可以完全替换现有标签集，也可以使用 GetJobTagging 检索现有标签集并在现有标签集中进行更改，修改该标签集，然后使用此 API 操作来将此标签集替换为您已修改的标签集。

    > **Note**
    >
    > 如果您发送带有空标签集的此请求，S3 分批操作将删除对象上的现有标签集。如果您使用此方法，则需为套餐 1 请求 (PUT) 付费。有关更多信息，请参阅 Amazon S3 定价。
    > 要删除分批操作任务的现有标签，首选执行 `DeleteJobTagging` 操作，因为该操作在不产生费用的情况下实现相同的结果。

- DeleteJobTagging – 删除与分批操作任务关联的标签集。

使用用于标记的任务标签创建分批操作任务

您可以通过添加标签来标记和控制对 S3 分批操作任务的访问。标签可用于确定谁负责分批操作任务。您可以创建附加了标签的任务，并且可以在创建任务后向任务添加标签。有关更多信息，请参阅 the section called “使用标签”。

使用 AWS CLI

以下 AWS CLI 示例创建了 S3 分批操作 `S3PutObjectCopy` 任务，其中使用任务标签作为该任务的标签。

1. 选择您希望分批操作任务执行的操作或 `OPERATION`，然后选择您的 `TargetResource`。

```
read -d '' OPERATION <<EOF
{
  "S3PutObjectCopy": {
    "TargetResource": "arn:aws:s3:::destination-bucket"
  }
```

```
}
EOF
```

2. 确定您需要用于此任务的任务 TAGS。在这种情况下，您应用两个标签 department 和 FiscalYear，值分别为 Marketing 和 2020。

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

3. 为分批操作任务指定 MANIFEST。

```
read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "EXAMPLE_S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::example-bucket/example_manifest.csv",
    "ETag": "example-5dc7a8bfb90808fc5d546218"
  }
}
EOF
```

4. 为分批操作任务配置 REPORT。

```
read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::example-report-bucket",
  "Format": "Example_Report_CSV_20180820",
```

```
  "Enabled": true,
  "Prefix": "reports/copy-with-replace-metadata",
  "ReportScope": "AllTasks"
}
EOF
```

5. 运行 `create-job` 操作以使用在上述步骤中设置的输入创建分批操作任务。

```
aws \
    s3control create-job \
    --account-id 123456789012 \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn arn:aws:iam::123456789012:role/batch-operations-role \
    --tags "${TAGS//$'\n'/}" \
    --client-request-token "$(uuidgen)" \
    --region us-west-2 \
    --description "Copy with Replace Metadata";
```

使用适用于 Java 的 AWS 开发工具包

Example

以下示例使用 AWS SDK for Java 创建带有标签的 S3 分批操作任务。

```
public String createJob(final AWSS3ControlClient awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::example-manifest-bucket/
manifests/10_manifest.csv";
    final String manifestObjectVersionId = "example-5dc7a8bfb90808fc5d546218";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new
 JobManifestSpec().withFormat(JobManifestFormat.S3InventoryReport_CSV_20161130);

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);
```

```
    final String jobReportBucketArn = "arn:aws:s3:::example-report-bucket";
    final String jobReportPrefix = "example-job-reports";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final String lambdaFunctionArn = "arn:aws:lambda:us-
west-2:123456789012:function:example-function";

    final JobOperation jobOperation = new JobOperation()
            .withLambdaInvoke(new
 LambdaInvokeOperation().withFunctionArn(lambdaFunctionArn));

    final S3Tag departmentTag = new
 S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");

    final String roleArn = "arn:aws:iam::123456789012:role/example-batch-operations-
role";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Test lambda job")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withTags(departmentTag, fiscalYearTag)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

从 S3 分批操作任务中删除标签

您可以使用这些示例从分批操作任务中删除标签。

使用 AWS CLI

以下示例使用 AWS CLI 从分批操作任务中删除标签。

```
aws \
    s3control delete-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

删除分批操作任务的任务标签

Example

以下示例使用 AWS SDK for Java 删除 S3 分批操作任务的标签。

```
public void deleteJobTagging(final AWSS3ControlClient awss3ControlClient,
                             final String jobId) {
    final DeleteJobTaggingRequest deleteJobTaggingRequest = new
 DeleteJobTaggingRequest()
            .withJobId(jobId);

    final DeleteJobTaggingResult deleteJobTaggingResult =
                awss3ControlClient.deleteJobTagging(deleteJobTaggingRequest);
}
```

为现有 S3 分批操作任务放置任务标签

您可以使用 PutJobTagging 将任务标签添加到您现有的 S3 分批操作任务中。有关更多信息，请参阅以下示例。

使用 AWS CLI

以下是使用 `s3control put-job-tagging` 通过 AWS CLI 向 S3 分批操作任务添加任务标签的示例。

> **Note**
>
> 如果您发送带有空标签集的此请求，S3 分批操作将删除对象上的现有标签集。此外，如果您使用此方法，则需为套餐 1 请求 (PUT) 付费。有关更多信息，请参阅 Amazon S3 定价。
> 要删除分批操作任务的现有标签，首选执行 DeleteJobTagging 操作，因为该操作在不产生费用的情况下实现相同的结果。

1. 确定您需要用于此任务的任务 TAGS。在这种情况下，您应用两个标签 department 和 FiscalYear，值分别为 Marketing 和 2020。

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

2. 使用所需参数运行 put-job-tagging 操作。

```
aws \
    s3control put-job-tagging \
    --account-id 123456789012 \
    --tags "${TAGS//$'\n'/}" \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

使用适用于 Java 的 AWS 开发工具包

Example

以下示例使用 AWS SDK for Java 在 S3 分批操作任务中放置标签。

```
public void putJobTagging(final AWSS3ControlClient awss3ControlClient,
```

```
                            final String jobId) {
    final S3Tag departmentTag = new
 S3Tag().withKey("department").withValue("Marketing");
    final S3Tag fiscalYearTag = new S3Tag().withKey("FiscalYear").withValue("2020");

    final PutJobTaggingRequest putJobTaggingRequest = new PutJobTaggingRequest()
            .withJobId(jobId)
            .withTags(departmentTag, fiscalYearTag);

    final PutJobTaggingResult putJobTaggingResult =
 awss3ControlClient.putJobTagging(putJobTaggingRequest);
}
```

获取 S3 分批操作任务的标签

您可以使用 `GetJobTagging` 返回 S3 分批操作任务的标签。有关更多信息，请参阅以下示例。

使用 AWS CLI

以下示例使用 AWS CLI 获取分批操作任务的标签。

```
aws \
    s3control get-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1;
```

使用适用于 Java 的 AWS 开发工具包

Example

以下示例使用 AWS SDK for Java 获取 S3 分批操作任务的标签。

```
public List<S3Tag> getJobTagging(final AWSS3ControlClient awss3ControlClient,
                                 final String jobId) {
    final GetJobTaggingRequest getJobTaggingRequest = new GetJobTaggingRequest()
            .withJobId(jobId);

    final GetJobTaggingResult getJobTaggingResult =
            awss3ControlClient.getJobTagging(getJobTaggingRequest);

    final List<S3Tag> tags = getJobTaggingResult.getTags();
```

```
    return tags;
}
```

使用任务标签控制 S3 分批操作的权限

为了帮助您管理 S3 分批操作任务，您可以添加任务标签。借助任务标签，您可以控制对 S3 分批操作任务的访问，并强制在创建任何任务时应用标签。

您最多可以为每个分批操作任务应用 50 个任务标签。这允许您设置非常精细的策略，限制可编辑任务的用户集。任务标签可授予或限制用户取消任务、激活处于确认状态的任务或更改任务优先级的能力。此外，您可以强制将标签应用于所有新任务，并为标签指定允许的键值对。您可以使用相同的 IAM 策略语言表达所有这些条件。有关更多信息，请参阅《Service Authorization Reference》中的 Actions, resources, and condition keys for Amazon S3。

以下示例显示如何使用 S3 分批操作任务标签仅向用户授予创建和编辑在特定部门（例如，财务或合规性部门）内运行的任务的权限。您还可以根据与任务相关的开发阶段分配任务，例如 QA 或生产。

在此示例中，您在 AWS Identity and Access Management(IAM) 策略中使用 S3 分批操作任务标签向用户授予只能创建和编辑在其部门内运行的任务的权限。您可以根据与任务相关的开发阶段分配任务，例如 QA 或生产。

此示例使用以下部门，每个部门使用分批操作的方式不同：

- 财务
- 合规性
- 商业智能
- 工程

主题

- 通过向用户和资源分配标签来控制访问
- 按阶段标记分批操作任务并对任务优先级实施限制

通过向用户和资源分配标签来控制访问

在这种情况下，管理员正在使用基于属性的访问控制 (ABAC)。ABAC 是一种 IAM 授权策略，它通过向用户和 AWS 资源附加标签来定义权限。

为用户和任务分配以下部门标签之一：

键：值

- `department : Finance`
- `department : Compliance`
- `department : BusinessIntelligence`
- `department : Engineering`

> **Note**
>
> 任务标签键和值区分大小写。

使用 ABAC 访问控制策略，您可以授予“财务”部门中的用户在其部门内创建和管理 S3 批量操作任务的权限，方法是将标签 `department=Finance` 与其用户关联。

此外，您可以将托管策略附加到 IAM 用户，此策略允许公司中的任何用户在其各自部门内创建或修改 S3 分批操作任务。

此示例中的策略包括三个策略语句：

- 策略中的第一个语句允许用户创建分批操作任务，前提是任务创建请求包含与其各自部门匹配的任务标签。这使用 `"${aws:PrincipalTag/department}"` 语法表示，该语法在策略评估时被用户的部门标签所取代。当在请求 (`"aws:RequestTag/department"`) 中为部门标签提供的值与用户的部门匹配时，该条件将得到满足。
- 策略中的第二个语句允许用户更改任务的优先级或更新任务的状态，前提是用户正在更新的任务与用户的部门匹配。
- 第三个语句允许用户随时通过 `PutJobTagging` 请求更新分批操作任务的标签，只要 (1) 其部门标签被保留，并且 (2) 他们正在更新的任务位于其部门内。

```
{
      "Version": "2012-10-17",
      "Statement": [
            {
                  "Effect": "Allow",
                  "Action": "s3:CreateJob",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
```

```
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": [
                        "s3:UpdateJobPriority",
                        "s3:UpdateJobStatus"
            ],
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": "s3:PutJobTagging",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        }
    ]
}
```

按阶段标记分批操作任务并对任务优先级实施限制

所有 S3 分批操作任务都具有数字优先级，Amazon S3 使用此优先级来决定以何种顺序运行任务。对于此示例，您可以限制大多数用户可以分配给任务的最大优先级，并为有限的特权用户集保留更高的优先级范围，如下所示：

- QA 阶段优先级范围（低）：1-100

- 生产阶段优先级范围（高）：1-300

为此，请引入一个代表任务阶段 的新标签集：

键：值

- stage : QA
- stage : Production

在部门内创建和更新低优先级任务

除了基于部门的限制外，此策略还对 S3 分批操作任务创建和更新引入了两个新的限制：

- 它允许用户在其部门中创建或更新任务，而一个新条件要求任务包含标签 stage=QA。
- 它允许用户创建或更新任务的优先级，新的最大优先级为 100。

```
{
        "Version": "2012-10-17",
        "Statement": [
        {
        "Effect": "Allow",
        "Action": "s3:CreateJob",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:UpdateJobStatus"
        ],
        "Resource": "*",
```

```
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:UpdateJobPriority",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:ResourceTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:PutJobTagging",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department" : "${aws:PrincipalTag/department}",
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA",
                "aws:ResourceTag/stage": "QA"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:GetJobTagging",
        "Resource": "*"
    }
    ]
}
```

在部门内创建和更新高优先级任务

少数用户可能需要在 QA 或 生产 中创建高优先级任务的能力。为了满足这一需求，您可以创建一个根据上一节中的低优先级策略改写的托管策略。

该策略执行以下操作：

- 允许用户使用标签 stage=QA 或 stage=Production 在其部门中创建或更新任务。
- 允许用户创建或更新任务的优先级，最高为 300。

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
                "Effect": "Allow",
                "Action": "s3:CreateJob",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                },
                      "NumericLessThanEquals": {
                            "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": [
                      "s3:UpdateJobStatus"
            ],
                "Resource": "*",
                "Condition": {
                      "StringEquals": {
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
```

```
            }
        },
          {
                "Effect": "Allow",
                "Action": "s3:UpdateJobPriority",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:ResourceTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
                      "StringEquals": {
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                      "NumericLessThanEquals": {
                            "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": "s3:PutJobTagging",
                "Resource": "*",
                "Condition": {
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ],
                            "aws:ResourceTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                }
            }
```

```
        }
    ]
}
```

使用 S3 分批操作管理 S3 对象锁定

S3 对象锁定还可以对对象版本进依法保留。与设置保留期限相似，依法保留可防止对象版本被覆盖或删除。但是，依法保留没有关联的保留期限，在删除之前将一直有效。有关更多信息，请参阅 S3 对象锁定依法保留。

有关将 S3 分批操作与对象锁定结合使用，以便将依法保留一次性添加到多个 Amazon S3 对象的信息，请参阅以下部分。

主题

- 使用 S3 分批操作启用 S3 对象锁定
- 使用分批操作设置对象锁定保留
- 将 S3 分批操作与 S3 对象锁定保留合规模式结合使用
- 将 S3 分批操作与 S3 对象锁定保留监管模式结合使用
- 使用 S3 分批操作关闭 S3 对象锁定依法保留

使用 S3 分批操作启用 S3 对象锁定

您可以将 S3 分批操作与 S3 对象锁定结合使用，以便同时为许多 Amazon S3 对象管理保留或启用依法保留。您可以在清单中指定目标对象的列表并将该列表提交到分批操作以完成操作。有关更多信息，请参阅 the section called “对象锁定保留” 和 the section called “对象锁定依法保留”。

以下示例说明如何创建具有 S3 分批操作权限的 IAM 角色并更新角色权限以创建启用对象锁定的任务。在这些示例中，将任何变量值替换为符合您要求的变量值。您还必须有一个 CSV 清单来标识 S3 分批操作任务的对象。有关更多信息，请参阅 the section called “指定清单”。

使用 AWS CLI

1. 创建 IAM 角色并为 S3 分批操作分配运行权限。

   所有 S3 分批操作任务都必须执行此步骤。

   ```
   export AWS_PROFILE='aws-user'
   ```

```
read -d '' bops_trust_policy <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "batchoperations.s3.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
aws iam create-role --role-name bops-objectlock --assume-role-policy-document
 "${bops_trust_policy}"
```

2. 设置要运行的 S3 分批操作和 S3 对象锁定。

   在此步骤中，允许角色执行以下操作：

   a. 在包含您希望在其上运行分批操作的目标对象的 S3 存储桶上运行对象锁定。

   b. 读取清单 CSV 文件和对象所在的 S3 存储桶。

   c. 将 S3 分批操作任务的结果写入报告存储桶。

```
read -d '' bops_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "s3:GetObjectVersion",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{ReportBucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name object-lock-
permissions --policy-document "${bops_permissions}"
```

使用适用于 Java 的 AWS 开发工具包

以下示例说明如何创建具有 S3 分批操作权限的 IAM 角色并更新角色权限，以使用 AWS SDK for Java 创建启用对象锁定的任务。在代码中，将任何变量值替换为符合您的要求的变量值。您还必须有一个 CSV 清单来标识 S3 分批操作任务的对象。有关更多信息，请参阅 the section called “指定清单”。

请执行下列步骤：

1. 创建 IAM 角色并为 S3 分批操作分配运行权限。所有 S3 分批操作任务都必须执行此步骤。
2. 设置要运行的 S3 分批操作和 S3 对象锁定。

   允许角色执行以下操作：

   1. 在包含您希望在其上运行分批操作的目标对象的 S3 存储桶上运行对象锁定。
   2. 读取清单 CSV 文件和对象所在的 S3 存储桶。
   3. 将 S3 分批操作任务的结果写入报告存储桶。

```
public void createObjectLockRole() {
    final String roleName = "bops-object-lock";

    final String trustPolicy = "{" +
            "  \"Version\": \"2012-10-17\", " +
            "  \"Statement\": [ " +
            "    { " +
            "      \"Effect\": \"Allow\", " +
            "      \"Principal\": { " +
            "        \"Service\": [" +
            "          \"batchoperations.s3.amazonaws.com\"" +
            "        ]" +
            "      }, " +
            "      \"Action\": \"sts:AssumeRole\" " +
            "    } " +
            "  ]" +
            "}";

    final String bopsPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": \"s3:GetBucketObjectLockConfiguration\"," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket\"" +
            "            ]" +
            "        }," +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:GetObject\"," +
            "                \"s3:GetObjectVersion\"," +
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }," +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObject\"," +
```

```
            "                \"s3:GetBucketLocation\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ReportBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final CreateRoleRequest createRoleRequest = new CreateRoleRequest()
            .withAssumeRolePolicyDocument(bopsPermissions)
            .withRoleName(roleName);

    final CreateRoleResult createRoleResult = iam.createRole(createRoleRequest);

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bopsPermissions)
            .withPolicyName("bops-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

使用分批操作设置对象锁定保留

以下示例允许此规则为清单存储桶中的对象设置 S3 对象锁定保留。

您可以更新角色以包含 `s3:PutObjectRetention` 权限，以便对存储桶中的对象运行对象锁定保留。

使用 AWS CLI

```
export AWS_PROFILE='aws-user'

read -d '' retention_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": [
                "arn:aws:s3:::{{ManifestBucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name retention-permissions
 --policy-document "${retention_permissions}"
```

使用适用于 Java 的 AWS 开发工具包

```
public void allowPutObjectRetention() {
    final String roleName = "bops-object-lock";

    final String retentionPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObjectRetention\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(retentionPermissions)
            .withPolicyName("retention-permissions")
            .withRoleName(roleName);
```

```
    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

将 S3 分批操作与 S3 对象锁定保留合规模式结合使用

以下示例基于上述创建信任策略以及在对象上设置 S3 分批操作和 S3 对象锁定配置权限的示例而构建。此示例将保留模式设置为 COMPLIANCE，并将 `retain until date` 设置为 2025 年 1 月 1 日。它将创建一个任务，该任务以清单存储桶中的对象为目标，并在您标识的报告存储桶中报告结果。

使用 AWS CLI

Example 设置跨多个对象的保留合规性

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-01T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/compliance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
```

```
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-bops",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Set compliance retain-until to 1 Jul 2030";
```

Example 将 **COMPLIANCE** 模式的 **retain until date** 延长至 2025 年 1 月 15 日

以下示例将 COMPLIANCE 模式的 retain until date 延长至 2025 年 1 月 15 日。

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-15T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF
```

```
read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/compliance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-bops",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Extend compliance retention to 15 Jan 2025";
```

使用适用于 Java 的 AWS 开发工具包

Example 将保留模式设置为“合规”，并将“保留到期日”设置为 2025 年 1 月 1 日。

```
public String createComplianceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/compliance-objects-
manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/compliance-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date janFirst = format.parse("01/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
                            .withRetainUntilDate(janFirst)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
```

```
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Set compliance retain-until to 1 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

Example 扩展 **COMPLIANCE** 模式的 **retain until date**

以下示例将 COMPLIANCE 模式的 `retain until date` 延长至 2025 年 1 月 15 日。

```
public String createExtendComplianceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/compliance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/compliance-objects-bops";

    final JobReport jobReport = new JobReport()
```

```
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan15th = format.parse("15/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.COMPLIANCE)
                            .withRetainUntilDate(jan15th)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Extend compliance retention to 15 Jan 2025")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

将 S3 分批操作与 S3 对象锁定保留监管模式结合使用

以下示例基于上述创建信任策略以及设置 S3 分批操作和 S3 对象锁定配置权限的示例而构建。它说明如何跨多个对象应用 S3 对象锁定保留监管，`retain until date` 为 2025 年 1 月 30 日。它会创建一个分批操作任务，该任务使用清单存储桶，并在报告存储桶中报告结果。

使用 AWS CLI

Example 在多个对象中应用 S3 对象锁定保留监管，“保留到期日”为 2025 年 1 月 30 日

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-30T00:00:00",
      "Mode":"GOVERNANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/governance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucketT",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/governance-objects",
  "ReportScope": "AllTasks"
}
```

```
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Put governance retention";
```

Example 绕过跨多个对象的保留管理

以下示例基于上述创建信任策略以及设置 S3 分批操作和 S3 对象锁定配置权限的示例而构建。它说明如何绕过跨多个对象的保留监管，并创建一个分批操作任务，该任务使用清单存储桶并在报告存储桶中报告结果。

```
export AWS_PROFILE='aws-user'

read -d '' bypass_governance_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name bypass-governance-
permissions --policy-document "${bypass_governance_permissions}"

export AWS_PROFILE='aws-user'
```

```
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "BypassGovernanceRetention": true,
    "Retention": {
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/governance-objects-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::REPORT_BUCKET",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/bops-governance",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
```

```
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Remove governance retention";
```

使用适用于 Java 的 AWS 开发工具包

以下示例基于上述创建信任策略以及设置 S3 分批操作和 S3 对象锁定配置权限的示例而构建。它说明如何跨多个对象应用 S3 对象锁定保留监管，而 `retain until date` 设置为 2025 年 1 月 30 日。它会创建一个分批操作任务，该任务使用清单存储桶，并在报告存储桶中报告结果。

Example 在多个对象中应用 S3 对象锁定保留监管，“保留到期日”为 2025 年 1 月 30 日

```
public String createGovernanceRetentionJob(final AWSS3ControlClient awss3ControlClient)
 throws ParseException {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/governance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/governance-objects";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
```

```
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan30th = format.parse("30/01/2025");

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()
                            .withMode(S3ObjectLockRetentionMode.GOVERNANCE)
                            .withRetainUntilDate(jan30th)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Put governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

Example 绕过跨多个对象的保留管理

以下示例基于上述创建信任策略以及设置 S3 分批操作和 S3 对象锁定配置权限的示例而构建。它说明如何绕过跨多个对象的保留监管，并创建一个分批操作任务，该任务使用清单存储桶并在报告存储桶中报告结果。

```
public void allowBypassGovernance() {
    final String roleName = "bops-object-lock";

    final String bypassGovernancePermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
            "        {" +
```

```
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:BypassGovernanceRetention\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(bypassGovernancePermissions)
            .withPolicyName("bypass-governance-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
public String createRemoveGovernanceRetentionJob(final AWSS3ControlClient
 awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/governance-objects-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/bops-governance";
```

```
    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectRetention(new S3SetObjectRetentionOperation()
                    .withRetention(new S3Retention()));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Remove governance retention")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

使用 S3 分批操作关闭 S3 对象锁定依法保留

以下示例基于上述创建信任策略以及设置 S3 分批操作和 S3 对象锁定配置权限的示例而构建。它说明如何使用分批操作在对象上禁用对象锁定依法保留。

在该示例中，首先更新角色以授予 `s3:PutObjectLegalHold` 权限，再创建一个分批操作任务（该任务从清单中标识的对象中禁用（删除）依法保留，然后报告此情况）。

使用 AWS CLI

Example 更新角色以授予 **`s3:PutObjectLegalHold`** 权限

```
export AWS_PROFILE='aws-user'
```

```
read -d '' legal_hold_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectLegalHold"
            ],
            "Resource": [
                "arn:aws:s3:::ManifestBucket/*"
            ]
        }
    ]

EOF

aws iam put-role-policy --role-name bops-objectlock --policy-name legal-hold-
permissions --policy-document "${legal_hold_permissions}"
```

Example 关闭依法保留

以下示例禁用了依法保留。

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/bops-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectLegalHold": {
    "LegalHold": {
      "Status":"OFF"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
```

```
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::ManifestBucket/legalhold-object-manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/legalhold-objects-bops",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Turn off legal hold";
```

使用适用于 Java 的 AWS 开发工具包

Example 更新角色以授予 **s3:PutObjectLegalHold** 权限

```
public void allowPutObjectLegalHold() {
    final String roleName = "bops-object-lock";

    final String legalHoldPermissions = "{" +
            "    \"Version\": \"2012-10-17\"," +
            "    \"Statement\": [" +
```

```
            "        {" +
            "            \"Effect\": \"Allow\"," +
            "            \"Action\": [" +
            "                \"s3:PutObjectLegalHold\"" +
            "            ]," +
            "            \"Resource\": [" +
            "                \"arn:aws:s3:::ManifestBucket/*\"" +
            "            ]" +
            "        }" +
            "    ]" +
            "}";

    final AmazonIdentityManagement iam =
            AmazonIdentityManagementClientBuilder.defaultClient();

    final PutRolePolicyRequest putRolePolicyRequest = new PutRolePolicyRequest()
            .withPolicyDocument(legalHoldPermissions)
            .withPolicyName("legal-hold-permissions")
            .withRoleName(roleName);

    final PutRolePolicyResult putRolePolicyResult =
 iam.putRolePolicy(putRolePolicyRequest);
}
```

Example 关闭依法保留

如果要禁用依法保留，请使用以下示例。

```
public String createLegalHoldOffJob(final AWSS3ControlClient awss3ControlClient) {
    final String manifestObjectArn = "arn:aws:s3:::ManifestBucket/legalhold-object-
manifest.csv";
    final String manifestObjectVersionId = "15ad5ba069e6bbc465c77bf83d541385";

    final JobManifestLocation manifestLocation = new JobManifestLocation()
            .withObjectArn(manifestObjectArn)
            .withETag(manifestObjectVersionId);

    final JobManifestSpec manifestSpec =
            new JobManifestSpec()
                    .withFormat(JobManifestFormat.S3BatchOperations_CSV_20180820)
                    .withFields("Bucket", "Key");

    final JobManifest manifestToPublicApi = new JobManifest()
            .withLocation(manifestLocation)
```

```
            .withSpec(manifestSpec);

    final String jobReportBucketArn = "arn:aws:s3:::ReportBucket";
    final String jobReportPrefix = "reports/legalhold-objects-bops";

    final JobReport jobReport = new JobReport()
            .withEnabled(true)
            .withReportScope(JobReportScope.AllTasks)
            .withBucket(jobReportBucketArn)
            .withPrefix(jobReportPrefix)
            .withFormat(JobReportFormat.Report_CSV_20180820);

    final JobOperation jobOperation = new JobOperation()
            .withS3PutObjectLegalHold(new S3SetObjectLegalHoldOperation()
                    .withLegalHold(new S3ObjectLockLegalHold()
                            .withStatus(S3ObjectLockLegalHoldStatus.OFF)));

    final String roleArn = "arn:aws:iam::123456789012:role/bops-object-lock";
    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = new CreateJobRequest()
            .withAccountId("123456789012")
            .withDescription("Turn off legal hold")
            .withManifest(manifestToPublicApi)
            .withOperation(jobOperation)
            .withPriority(priority)
            .withRoleArn(roleArn)
            .withReport(jobReport)
            .withConfirmationRequired(requiresConfirmation);

    final CreateJobResult result = awss3ControlClient.createJob(request);

    return result.getJobId();
}
```

S3 分批操作教程

以下教程提供了一些分批操作任务的完整端到端过程。

- 教程：使用 S3 批量操作、AWS Lambda 和 AWS Elemental MediaConvert 对视频进行批量转码

监控 Amazon S3

监控是保持 Amazon S3 和您的 AWS 解决方案的可靠性、可用性和性能的重要方面。我们推荐您从 AWS 解决方案的所有方面收集监控数据，以便更轻松地调试出现的多点故障。在开始监控 Amazon S3 之前，创建一个监控计划，解决以下问题：

- 监控目的是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

有关 Amazon S3 中的日志记录和监控的更多信息，请参阅以下主题。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

主题

监控工具

AWS 为您提供了各种可用于监控 Amazon S3 的工具。您可以配置其中的一些工具来为您执行监控任务，但有些工具需要手动干预。建议您尽可能实现监控任务自动化。

自动监控工具

您可以使用以下自动化监控工具来监控 Amazon S3，并在出现错误时进行报告：

- Amazon CloudWatch 警报 – 按您指定的时间段观察单个指标，并根据相对于给定阈值的指标值在若干时间段内执行一项或多项操作。操作是一个发送到 Amazon Simple Notification Service (Amazon SNS) 主题或 Amazon EC2 Auto Scaling 策略的通知。CloudWatch 警报将不会调用操作，因为这些操作处于特定状态。该状态必须已改变并在指定的若干个时间段内保持不变。有关更多信息，请参阅使用 Amazon CloudWatch 监控指标。
- AWS CloudTrail 日志监控 – 在账户间共享日志文件，通过将 CloudTrail 日志文件发送到 CloudWatch Logs 对它们进行实时监控，在 Java 中编写日志处理应用程序，以及验证您的日志文件在由 CloudTrail 交付后未发生更改。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon S3 API 调用。

手动监控工具

监控 Amazon S3 的另一个重要环节是手动监控 CloudWatch 警报未涵盖的那些项。Amazon S3、CloudWatch、Trusted Advisor 和其他 AWS Management Console 控制面板提供您的 AWS 环境状态的概览视图。您可能要启用服务器访问日志记录，以跟踪针对存储桶的访问请求。每个访问日志记录都提供有关单个访问请求的详细信息，如请求者、存储桶名称、请求时间、请求操作、响应状态和错误代码（如果有）。有关更多信息，请参阅 使用服务器访问日志记录来记录请求。

- Amazon S3 控制面板显示以下内容：
  - 您的存储桶及其包含的对象和属性
- CloudWatch 主页显示以下内容：
  - 当前警报和状态
  - 告警和资源图表
  - 服务运行状况

  此外，您还可以使用 CloudWatch 执行以下操作：
  - 创建自定义控制面板以监控您关心的服务。
  - 绘制指标数据图，以排除问题并弄清楚趋势。
  - 搜索并浏览您所有的 AWS 资源指标。
  - 创建和编辑警报以接收有关问题的通知。

- AWS Trusted Advisor 可以帮助您监控 AWS 资源以提高性能、可靠性、安全性和成本效益。四个 Trusted Advisor 检查可供所有用户使用；超过 50 个检查可供具有“商业”或“企业”支持计划的用户使用。有关更多信息，请参阅 AWS Trusted Advisor。

  Trusted Advisor 具有与 Amazon S3 相关的以下检查：

  - 检查 Amazon S3 存储桶的日志记录配置。
  - 具有开放访问权限的 Amazon S3 存储桶的安全性检查。
  - 未启用版本控制或已暂停版本控制的 Amazon S3 存储桶的容错检查。

Amazon S3 的日志记录选项

您可以记录用户、角色或 AWS 服务对 Amazon S3 资源所采取的操作，并维护日志记录以满足审计和合规性目的。为此，您可以使用服务器访问日志记录、AWS CloudTrail 日志记录，或两者的组合。我们建议您使用 CloudTrail 为您的 Amazon S3 资源记录存储桶级和对象级操作的日志。请参阅下面几节，了解有关每个选项的更多信息：

- 使用服务器访问日志记录来记录请求
- 使用 AWS CloudTrail 记录 Amazon S3 API 调用

下表列出了 CloudTrail 日志和 Amazon S3 服务器访问日志的关键属性。要确保 CloudTrail 满足您的安全要求，请查看表和注释。

日志属性	AWS CloudTrail	Amazon S3 服务器日志
可以转发到其他系统（Amazon CloudWatch Logs、Amazon CloudWatch Events）	是	No
将日志传输到多个目标（例如，将相同的日志发送到两个不同的存储桶）	是	No
对对象的子级开启日志（前缀）	是	No

日志属性	AWS CloudTrail	Amazon S3 服务器日志
跨账户日志传输（不同账户拥有的目标和源存储桶）	是	No
使用数字签名或哈希对日志文件进行完整性验证	是	No
默认或选择加密日志文件	是	No
对象操作（使用 Amazon S3 API）	是	是
存储桶操作（使用 Amazon S3 API）	是	是
日志的可搜索 UI	是	No
对象锁定参数的字段，Amazon S3 选择日志记录的属性	是	No
日志记录的 `Object Size`、`Total Time`、`Turn-Around Time` 和 `HTTP Referer` 的字段	否	是
生命周期转换，过期，还原	否	是
批处理删除操作中键的日志记录	否	是
身份验证失败[1]	否	是
日志得到传输的账户	存储桶拥有者[2] 以及请求者	仅限存储桶拥有者
Performance and Cost	AWS CloudTrail	Amazon S3 Server Logs
价格	管理事件（第一次传输）免费；数据事件引发费用，此外还有日志存储	除日志存储之外，没有其他费用

日志属性	AWS CloudTrail	Amazon S3 服务器日志
日志传输的速度	每 5 分钟传输数据事件；每 15 分钟传输管理事件	几个小时内
日志格式	JSON	具有以空格分隔、新行分隔的记录的日志文件

注意

1. CloudTrail 对于未通过身份验证（其中，提供的凭证无效）的请求不传输日志。但是，对于授权失败的请求 (`AccessDenied`) 和匿名用户发出的请求，它的确包括日志。
2. 当账户对请求中的此对象没有完全访问权限时，S3 存储桶拥有者将接收 CloudTrail 日志。有关更多信息，请参阅 跨账户情形中的 Amazon S3 对象级操作。
3. 对于 VPC 端点策略拒绝的 VPC 端点请求，或者对于在评估 VPC 策略之前失败的请求，S3 不支持向请求者或存储桶拥有者传输 CloudTrail 日志或服务器访问日志。

使用 AWS CloudTrail 记录 Amazon S3 API 调用

Amazon S3 与 AWS CloudTrail 集成，后者是一项提供用户、角色或 AWS 服务所采取操作的记录的服务。CloudTrail 将 Amazon S3 的所有 API 调用作为事件捕获。捕获的调用中包括通过 Amazon S3 控制台的调用和对 Amazon S3 API 操作的代码调用。借助通过 CloudTrail 收集的信息，您可以确定向 Amazon S3 发出哪些请求、发出请求的 IP 地址、请求的发出时间以及其它详细信息。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是否代表 IAM Identity Center 用户发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务 发出。

当您创建账户并可以自动访问 CloudTrail 事件历史记录时，CloudTrail 在您的 AWS 账户 中处于活动状态。CloudTrail 事件历史记录提供对 AWS 区域 中过去 90 天的已记录管理事件的可查看、可搜索、可下载和不可变记录。有关更多信息，请参见《AWS CloudTrail 用户指南》的 使用 CloudTrail 事件历史记录。查看事件历史记录不会收取 CloudTrail 费用。

要持续记录您的 AWS 账户 过去 90 天的事件，请创建跟踪或 CloudTrail Lake 事件数据存储。

CloudTrail 跟踪

通过跟踪记录，CloudTrail 可将日志文件传送至 Simple Storage Service (Amazon S3) 存储桶。使用 AWS Management Console 创建的所有跟踪均具有多区域属性。您可以通过使用 AWS CLI 创建单区域或多区域跟踪。建议创建多区域跟踪，因为您可记录您账户中的所有 AWS 区域 的活动。如果您创建单区域跟踪，则只能查看跟踪的 AWS 区域 中记录的事件。有关跟踪的更多信息，请参阅《AWS CloudTrail 用户指南》中的为您的 AWS 账户 创建跟踪和为组织创建跟踪。

通过创建跟踪，您可以从 CloudTrail 免费向您的 Amazon S3 存储桶传送一份正在进行的管理事件的副本，但会收取 Amazon S3 存储费用。有关 CloudTrail 定价的更多信息，请参阅 AWS CloudTrail 定价。有关 Amazon S3 定价的信息，请参阅 Amazon S3 定价。

CloudTrail Lake 事件数据存储

CloudTrail Lake 允许您对事件运行基于 SQL 的查询。CloudTrail Lake 可将基于行的 JSON 格式的现有事件转换为 Apache ORC 格式。ORC 是一种针对快速检索数据进行优化的列式存储格式。事件将被聚合到事件数据存储中，它是基于您通过应用高级事件选择器选择的条件的不可变的事件集合。应用于事件数据存储的选择器用于控制哪些事件持续存在并可供您查询。有关 CloudTrail Lake 的更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 AWS CloudTrail Lake。

CloudTrail Lake 事件数据存储和查询会产生费用。创建事件数据存储时，您可以选择要用于事件数据存储的定价选项。定价选项决定了摄取和存储事件的成本，以及事件数据存储的默认和最长保留期。有关 CloudTrail 定价的更多信息，请参阅 AWS CloudTrail 定价。

日志文件可以在存储桶中存储任意长时间，不过您也可以定义 Amazon S3 生命周期规则以自动归档或删除日志文件。默认情况下，将使用 Amazon S3 服务器端加密 (SSE) 对日志文件进行加密。

将 CloudTrail 日志与 Amazon S3 服务器访问日志和 CloudWatch Logs 结合使用

AWS CloudTrail 日志记录用户、角色或 AWS 服务在 Amazon S3 中执行的操作，而 Amazon S3 服务器访问日志提供向 S3 存储桶发出的请求的详细记录。有关不同日志的工作方式及其属性、性能和成本的更多信息，请参阅 the section called “日志记录选项”。

您可以将 AWS CloudTrail 日志与 Amazon S3 的服务器访问日志搭配使用。CloudTrail 日志为您提供了 Amazon S3 存储桶级别和对象级操作的详细 API 跟踪。Amazon S3 的服务器访问日志可让您了解对于 Amazon S3 中您的数据的对象级操作。有关服务器访问日志的更多信息，请参阅 使用服务器访问日志记录来记录请求。

您还可以将 CloudTrail 日志与 Amazon S3 的 Amazon CloudWatch 一起使用。CloudTrail 与 CloudWatch Logs 集成可将 CloudTrail 捕获到的 S3 存储桶级 API 活动传送给您指定的 CloudWatch 日志组中的 CloudWatch 日志流。您可以创建用于监控特定 API 活动的 CloudWatch 警报，并在此特定 API 活动发生时收到电子邮件通知。有关用于监控特定 API 活动的 CloudWatch 警报的更多信息，请参阅 AWS CloudTrail 用户指南。有关将 CloudWatch 与 Amazon S3 结合使用的更多信息，请参阅使用 Amazon CloudWatch 监控指标。

> **Note**
>
> 当 VPC 端点策略拒绝时，S3 不支持向 VPC 端点请求的请求者或存储桶拥有者传输 CloudTrail 日志。

使用 Amazon S3 SOAP API 调用进行 CloudTrail 跟踪

CloudTrail 跟踪 Amazon S3 SOAP API 调用。Amazon S3 HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。有关 Amazon S3 SOAP 支持的更多信息，请参阅 附录 A：使用 SOAP API。

> **Important**
>
> SOAP 不支持较新的 Amazon S3 特征。我们建议您使用 REST API 或 AWS SDK。

CloudTrail 日志记录跟踪的 Amazon S3 SOAP 操作

SOAP API 名称	CloudTrail 日志中使用的 API 事件名称
ListAllMyBuckets	`ListBuckets`
CreateBucket	`CreateBucket`
DeleteBucket	`DeleteBucket`
GetBucketAccessControlPolicy	`GetBucketAcl`
SetBucketAccessControlPolicy	`PutBucketAcl`
GetBucketLoggingStatus	`GetBucketLogging`
SetBucketLoggingStatus	`PutBucketLogging`

有关 CloudTrail 和 Amazon S3 的更多信息，请参阅以下主题：

主题

Amazon S3 CloudTrail 事件

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

本节提供有关 S3 记录到 CloudTrail 的事件的信息。

CloudTrail 中的 Amazon S3 数据事件

数据事件可提供对资源或在资源中所执行资源操作（例如，读取或写入 Amazon S3 对象）的相关信息。这些也称为数据层面操作。数据事件通常是高容量活动。默认情况下，CloudTrail 不记录数据事件。CloudTrail 事件历史记录不记录数据事件。

记录数据事件将收取额外费用。有关 CloudTrail 定价的更多信息，请参阅 AWS CloudTrail 定价。

您可以使用 CloudTrail 控制台、AWS CLI 或 CloudTrail API 操作记录 Amazon S3 资源类型的数据事件。有关如何记录数据事件的更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 AWS Management Console 记录数据事件和使用 AWS Command Line Interface 记录数据事件。

下表列出了您可以为其记录数据事件的 Amazon S3 资源类型。数据事件类型（控制台）列显示可从 CloudTrail 控制台上的数据事件类型列表中选择的值。resources.type 值列显示了您在使用 AWS CLI 或 CloudTrail API 配置高级事件选择器时需要指定的 `resources.type` 值。记录到 CloudTrail 的数据 API 列显示了针对该资源类型记录到 CloudTrail 的 API 调用。

数据事件类型（控制台）	resources.type 值	记录至 CloudTrail 的数据 API
S3	`AWS::S3::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • DeleteObjects • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • GetObjectTorrent • HeadObject • ListMultipartUploads • ListObjectVersions • ListObjects • ListParts • PutObject • PutObjectAcl • PutObjectLegalHold • PutObjectRetention • PutObjectTagging • RestoreObject • SelectObjectContent • UploadPart • UploadPartCopy

数据事件类型（控制台）	resources.type 值	记录至 CloudTrail 的数据 API
S3 Express One Zone	`AWS::S3Express::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CreateSession • CopyObject • CreateMultipartUpload • DeleteObject • DeleteObjects • GetObject • GetObjectAttributes • HeadBucket • HeadObject • ListObjectsV2 • ListParts • PutObject • UploadPart • UploadPartCopy

数据事件类型（控制台）	resources.type 值	记录至 CloudTrail 的数据 API
S3 接入点	`AWS::S3::Access Point`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject（仅限同区域副本） • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetBucketAcl • GetBucketCors • GetBucketLocation • GetBucketNotificationConfiguration • GetBucketPolicy • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadBucket • HeadObject • ListMultipartUploads • ListObjects • ListObjectsV2 • ListObjectVersions • ListParts • Presign • PutObject • PutObjectLegalHold

数据事件类型（控制台）	resources.type 值	记录至 CloudTrail 的数据 API
		• PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart • UploadPartCopy（仅限同区域副本）

数据事件类型（控制台）	resources.type 值	记录至 CloudTrail 的数据 API
S3 对象 Lambda	`AWS::S3ObjectLambda::AccessPoint`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject（仅限同区域副本） • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectAcl • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadObject • ListMultipartUploads • ListObjects • ListObjectVersions • ListParts • PutObject • PutObjectLegalHold • PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart • WriteGetObjectResponse

数据事件类型（控制台）	resources.type 值	记录至 CloudTrail 的数据 API
S3 Outposts	`AWS::S3Outposts::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject（仅限同区域副本） • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectTagging • HeadObject • ListMultipartUploads • ListObjects • ListObjectsV2 • ListParts • PutObject • PutObjectTagging • UploadPart • UploadPartCopy

您可以将高级事件选择器配置为在 `eventName`、`readOnly` 和 `resources.ARN` 字段上进行筛选，从而仅记录那些对您很重要的事件。有关这些字段的更多信息，请参阅《AWS CloudTrail API 参考》中的 AdvancedFieldSelector。

CloudTrail 中的 Amazon S3 管理事件

Amazon S3 将所有控制面板操作记录为管理事件。有关 S3 API 操作的更多信息，请参阅 Amazon S3 API Reference。

CloudTrail 如何捕获向 Amazon S3 发出的请求

默认情况下，CloudTrail 会记录在过去 90 天内进行的 S3 存储桶级别 API 调用，但不会记录向对象发出的请求。存储桶级调用包括诸如

`CreateBucket`、`DeleteBucket`、`PutBucketLifecycle`、`PutBucketPolicy` 等事件。您可以在 CloudTrail 控制台上看到存储桶级事件。但是，您无法在此查看数据事件（Amazon S3 对象级调用），您必须解析或查询它们的 CloudTrail 日志。

CloudTrail 日志记录跟踪的 Amazon S3 账户级操作

CloudTrail 会记录账户级操作。Amazon S3 记录与其他 AWS 服务记录一起写入日志文件。CloudTrail 基于时间段和文件大小来确定何时创建并写入新文件。

本节中的表格列出了可用于 CloudTrail 日志记录的 Amazon S3 账户级操作。

CloudTrail 日志记录跟踪的 Amazon S3 账户级 API 操作显示为以下事件名称。CloudTrail 事件名称与 API 操作名称不同。例如，DeletePublicAccessBlock 是 DeleteAccountPublicAccessBlock。

- DeleteAccountPublicAccessBlock
- GetAccountPublicAccessBlock
- PutAccountPublicAccessBlock

CloudTrail 日志记录跟踪的 Amazon S3 存储桶级操作

默认情况下，CloudTrail 将记录通用存储桶的存储桶级操作。Amazon S3 记录与其他 AWS 服务记录一起写入日志文件。CloudTrail 基于时间段和文件大小来确定何时创建并写入新文件。

此部分列出了可用于 CloudTrail 日志记录的 Amazon S3 存储桶级操作。

CloudTrail 日志记录跟踪的 Amazon S3 存储桶级 API 操作显示为以下事件名称。在某些情况下，CloudTrail 事件名称与 API 操作名称不同。例如，`PutBucketLifecycleConfiguration` 为 `PutBucketLifecycle`。

- CreateBucket
- DeleteBucket
- DeleteBucketAnalyticsConfiguration
- DeleteBucketCors
- DeleteBucketEncryption
- DeleteBucketIntelligentTieringConfiguration
- DeleteBucketInventoryConfiguration

- DeleteBucketLifecycle
- DeleteBucketMetricsConfiguration
- DeleteBucketOwnershipControls
- DeleteBucketPolicy
- DeleteBucketPublicAccessBlock
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccelerateConfiguration
- GetBucketAcl
- GetBucketAnalyticsConfiguration
- GetBucketCors
- GetBucketEncryption
- GetBucketIntelligentTieringConfiguration
- GetBucketInventoryConfiguration
- GetBucketLifecycle
- GetBucketLocation
- GetBucketLogging
- GetBucketMetricsConfiguration
- GetBucketNotification
- GetBucketObjectLockConfiguration
- GetBucketOwnershipControls
- GetBucketPolicy
- GetBucketPolicyStatus
- GetBucketPublicAccessBlock
- GetBucketReplication
- GetBucketRequestPayment
- GetBucketTagging
- GetBucketVersioning
- GetBucketWebsite

- HeadBucket
- ListBuckets
- PutAccelerateConfiguration
- PutBucketAcl
- PutBucketAnalyticsConfiguration
- PutBucketCors
- PutBucketEncryption
- PutBucketIntelligentTieringConfiguration
- PutBucketInventoryConfiguration
- PutBucketLifecycle
- PutBucketLogging
- PutBucketMetricsConfiguration
- PutBucketNotification
- PutBucketObjectLockConfiguration
- PutBucketOwnershipControls
- PutBucketPolicy
- PutBucketPublicAccessBlock
- PutBucketReplication
- PutBucketRequestPayment
- PutBucketTagging
- PutBucketVersioning
- PutBucketWebsite

除了这些 API 操作之外，还可以使用 OPTIONS object 对象级操作。由于此操作将检查存储桶的 CORS 配置，因此它被视为 CloudTrail 日志记录中的存储桶级操作。

CloudTrail 日志记录跟踪的 Amazon S3 Express One Zone 存储类存储桶级别（区域 API 端点）操作

默认情况下，CloudTrail 将目录存储桶的存储桶级操作记录为管理事件。S3 Express One Zone 的 CloudTrail 管理事件的 `eventsource` 为 `s3express.amazonaws.com`。

以下区域端点 API 操作将记录到 CloudTrail 中。

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- PutBucketPolicy
- ListDirectoryBuckets

有关更多信息，请参阅使用 AWS CloudTrail 为 S3 Express One Zone 记录日志

跨账户情形中的 Amazon S3 对象级操作

下面是涉及跨账户情形中的对象级 API 调用的特殊使用案例以及报告 CloudTrail 日志的方法。CloudTrail 会将日志提供给申请方（进行 API 调用的账户），除非在某些拒绝访问的情况下，其中日志条目被编辑或省略。在设置跨账户访问时，请考虑本部分中的示例。

> **Note**
>
> 这些示例假定 CloudTrail 日志已适当配置。

示例 1：CloudTrail 将日志提供给存储桶拥有者

即使存储桶拥有者不具有执行相同对象 API 操作的权限，CloudTrail 也会将日志传递给存储桶拥有者。请考虑以下跨账户情形：

- 账户 A 拥有该存储桶。
- 账户 B（请求者）尝试访问该存储桶中的对象。
- 账户 C 拥有该对象。账户 C 可能与账户 A 是同一个账户，也可能不是同一个账户。

> **Note**
>
> CloudTrail 始终将对象级 API 日志提供给请求者（账户 B）。此外，即使存储桶拥有者不拥有该对象（账户 C）或对该对象不具有执行相同 API 操作的权限，CloudTrail 也会向存储桶拥有者（账户 A）提供相同的日志。

示例 2：CloudTrail 没有使设置对象 ACL 时使用的电子邮件地址激增

请考虑以下跨账户情形：

- 账户 A 拥有该存储桶。
- 账户 B（请求者）使用电子邮件地址发送了一个设置对象 ACL 授权的请求。有关 ACL 的更多信息，请参阅 访问控制列表 (ACL) 概述。

该请求方将获取日志以及电子邮件信息。但是，存储桶拥有者（如果他们像在示例 1 中一样有资格接收日志）将获取报告该事件的 CloudTrail 日志。但是，存储桶拥有者不会获取 ACL 配置信息，尤其是被授权者电子邮件地址和授权。日志为存储桶拥有者提供的唯一信息是账户 B 进行了 ACL API 调用。

Amazon S3 和 S3 on Outposts 的 CloudTrail 日志文件条目

Important

Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

一个事件表示一个来自任何源的请求，包括有关所请求的 API 操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此事件不会按任何特定顺序显示。

Note

要查看 Amazon S3 Express One Zone 存储类的 CloudTrail 日志文件示例，请参阅 CloudTrail log file examples for S3 Express One Zone。

有关更多信息，请参阅以下示例。

主题

- 示例：Amazon S3 的 CloudTrail 日志文件条目

- 示例：Amazon S3 on Outposts 日志文件条目

示例：Amazon S3 的 CloudTrail 日志文件条目

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 GET 服务、PutBucketAcl 和 GetBucketVersioning 操作。


```
{
    "Records": [
    {
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
        "eventTime": "2019-02-01T03:18:19Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "ListBuckets",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "[]",
        "requestParameters": {
            "host": [
                "s3.us-west-2.amazonaws.com"
            ]
        },
        "responseElements": null,
        "additionalEventData": {
            "SignatureVersion": "SigV2",
            "AuthenticationMethod": "QueryString",
            "aclRequired": "Yes"
    },
        "requestID": "47B8E8D397DCE7A6",
        "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",
        "eventType": "AwsApiCall",
        "recipientAccountId": "444455556666",
        "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
```

```
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    },
    {
       "eventVersion": "1.03",
       "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
      "eventTime": "2019-02-01T03:22:33Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "PutBucketAcl",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "bucketName": "",
          "AccessControlPolicy": {
              "AccessControlList": {
                  "Grant": {
                      "Grantee": {
                          "xsi:type": "CanonicalUser",
                          "xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
                          "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
                       },
                      "Permission": "FULL_CONTROL"
                   }
              },
              "xmlns": "http://s3.amazonaws.com/doc/2006-03-01/",
              "Owner": {
                  "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
              }
          },
          "host": [
              "s3.us-west-2.amazonaws.com"
          ],
          "acl": [
              ""
```

```
            ]
        },
        "responseElements": null,
        "additionalEventData": {
            "SignatureVersion": "SigV4",
            "CipherSuite": "ECDHE-RSA-AES128-SHA",
            "AuthenticationMethod": "AuthHeader"
        },
        "requestID": "BD8798EACDD16751",
        "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
        "eventType": "AwsApiCall",
        "recipientAccountId": "111122223333",
        "tlsDetails": {
              "tlsVersion": "TLSv1.2",
              "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
              "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    },
    {
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
          },
        "eventTime": "2019-02-01T03:26:37Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "GetBucketVersioning",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "",
        "userAgent": "[]",
        "requestParameters": {
            "host": [
                "s3.us-west-2.amazonaws.com"
            ],
            "bucketName": "amzn-s3-demo-bucket1",
            "versioning": [
                ""
            ]
        },
        "responseElements": null,
```

```
      "additionalEventData": {
          "SignatureVersion": "SigV4",
          "CipherSuite": "ECDHE-RSA-AES128-SHA",
          "AuthenticationMethod": "AuthHeader"
    },
      "requestID": "07D681279BD94AED",
      "eventID": "f2b287f3-0df1-4961-a2f4-c4bdfed47657",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333",
      "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    }
  ]
}
```

示例：Amazon S3 on Outposts 日志文件条目

Amazon S3 on Outposts 管理事件可通过 AWS CloudTrail 提供。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon S3 API 调用。此外，您还可以选择性地为 AWS CloudTrail 中的数据事件启用日志记录。

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的区域的 S3 存储桶中。Outposts 存储桶的 CloudTrail 日志包括一个新字段 `edgeDeviceDetails`，该字段用于识别指定存储桶所在的 Outpost。

其他日志字段包括请求的操作、操作的日期和时间以及请求参数。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `s3-outposts` 上的 PutObject 操作。

```
{
      "eventVersion": "1.08",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/yourUserName",
        "accountId": "222222222222",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "yourUserName"
      },
```

```
      "eventTime": "2020-11-30T15:44:33Z",
      "eventSource": "s3-outposts.amazonaws.com",
      "eventName": "PutObject",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "26.29.66.20",
      "userAgent": "aws-cli/1.18.39 Python/3.4.10 Darwin/18.7.0 botocore/1.15.39",
      "requestParameters": {
        "expires": "Wed, 21 Oct 2020 07:28:00 GMT",
        "Content-Language": "english",
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
        "ObjectCannedACL": "BucketOwnerFullControl",
        "x-amz-server-side-encryption": "Aes256",
        "Content-Encoding": "gzip",
        "Content-Length": "10",
        "Cache-Control": "no-cache",
        "Content-Type": "text/html; charset=UTF-8",
        "Content-Disposition": "attachment",
        "Content-MD5": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
        "x-amz-storage-class": "Outposts",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "bucketName": "amzn-s3-demo-bucket1",
        "Key": "path/upload.sh"
      },
      "responseElements": {
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
        "x-amz-server-side-encryption": "Aes256",
        "x-amz-version-id": "001",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "ETag": "d41d8cd98f00b204e9800998ecf8427f"
      },
      "additionalEventData": {
        "CipherSuite": "ECDHE-RSA-AES128-SHA",
        "bytesTransferredIn": 10,
        "x-amz-id-2": "29xXQBV20
+xOHKItvzY1suLv1i6A52E0zOX159fpfsItYd58JhXwKxXAXI4IQkp6",
        "SignatureVersion": "SigV4",
        "bytesTransferredOut": 20,
        "AuthenticationMethod": "AuthHeader"
      },
      "requestID": "8E96D972160306FA",
      "eventID": "ee3b4e0c-ab12-459b-9998-0a5a6f2e4015",
      "readOnly": false,
```

```
      "resources": [
        {
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Object",
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/path/upload.sh"
        },
        {
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Bucket",
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/"
        }
      ],
      "eventType": "AwsApiCall",
      "managementEvent": false,
      "recipientAccountId": "444455556666",
      "sharedEventID": "02759a4c-c040-4758-b84b-7cbaaf17747a",
      "edgeDeviceDetails": {
        "type": "outposts",
        "deviceId": "op-01ac5d28a6a232904"
      },
      "eventCategory": "Data"
    }
```

为 S3 存储桶和对象启用 CloudTrail 事件日志记录

您可以使用 CloudTrail 数据事件获取有关 Amazon S3 中存储桶和对象级别请求的信息。要为您的所有存储桶或特定存储桶列表启用 CloudTrail 数据事件，您必须在 CloudTrail 中手动创建跟踪记录。

> **Note**
>
> - CloudTrail 的默认设置是仅查找管理事件。请检查以确保已为您的账户启用数据事件。
> - 利用正在生成高工作负载的 S3 存储桶，您可以在很短时间内快速生成数千条日志。请注意您为某个繁忙存储桶启用 CloudTrail 数据事件所选择的时长。

CloudTrail 将 Amazon S3 数据事件日志存储在您选择的 S3 存储桶中。考虑在单独的 AWS 账户中使用一个存储桶将多个存储桶中您可能拥有的事件整理到一个集中位置，以便更轻松地进行查询和分析。AWS Organizations 可帮助您创建与拥有您所监控的存储桶的账户关联的 AWS 账户。有关更多信息，请参阅AWS Organizations《用户指南》中的什么是AWS Organizations？。

在 CloudTrail 中记录跟踪的数据事件时，您可以选择使用高级事件选择器或基本事件选择器，来记录存储在通用存储桶中的对象的数据事件。要记录存储在目录存储桶中的对象的数据事件，必须使用高级事件选择器。有关更多信息，请参阅使用 AWS CloudTrail 为 S3 Express One Zone 记录日志。

在 CloudTrail 控制台中使用高级事件选择器创建跟踪时，在数据事件部分中，您可以在记录选择器模板中选择记录所有事件，来记录所有对象级事件。在 CloudTrail 中使用基本事件选择器创建跟踪时，在数据事件部分中，您可以选中选择您账户中的所有 S3 存储桶复选框以记录所有对象级事件。

> **Note**
>
> - 最佳实践是为 AWS CloudTrail 数据事件存储桶创建生命周期配置。配置生命周期配置，以便在您认为需要审计日志文件的时间段之后定期删除这些日志文件。这样做可以减少 Athena 为每个查询分析的数据量。有关更多信息，请参阅 在存储桶上设置生命周期配置。
> - 有关日志记录格式的更多信息，请参阅使用 AWS CloudTrail 记录 Amazon S3 API 调用。
> - 有关如何查询 CloudTrail 日志的示例，请参阅 AWS 大数据博客文章使用 AWS CloudTrail 和 Amazon Athena 分析安全性、合规性和操作活动。

使用控制台为存储桶中的对象启用日志记录功能

您可以使用 Amazon S3 控制台配置 AWS CloudTrail 跟踪来记录 S3 存储桶中的对象的数据事件。CloudTrail 支持记录 Amazon S3 对象级别 API 操作，例如 `GetObject`、`DeleteObject` 和 `PutObject`。这些事件称为数据事件。

默认情况下，CloudTrail 跟踪不会记录数据事件，但您可以将跟踪配置为记录您指定的 S3 存储桶的数据事件，或记录 AWS 账户 中所有 Amazon S3 存储桶的数据事件。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon S3 API 调用。

CloudTrail 不会在 CloudTrail 事件历史记录中填充数据事件。此外，并非所有存储桶级别的操作都会填充在 CloudTrail 事件历史记录中。有关 CloudTrail 日志记录跟踪的 Amazon S3 存储桶级 API 操作的更多信息，请参阅CloudTrail 日志记录跟踪的 Amazon S3 存储桶级操作。有关如何查询 CloudTrail 日志的更多信息，请参阅关于使用 Amazon CloudWatch Logs 筛选条件模式和 Amazon Athena 查询 CloudTrail 日志的 AWS 知识中心文章。

要配置跟踪以记录某个 S3 存储桶的数据事件，您可以使用 AWS CloudTrail 控制台或 Amazon S3 控制台。如果您要配置跟踪以记录您的 AWS 账户 中所有 Amazon S3 存储桶的数据事件，使用 CloudTrail 控制台会更轻松。有关使用 CloudTrail 控制台配置跟踪以记录 S3 数据事件的信息，请参阅 AWS CloudTrail 用户指南中的数据事件。

> **Important**
>
> 记录数据事件将收取额外费用。有关更多信息，请参阅 AWS CloudTrail 定价。

以下过程演示如何使用 Amazon S3 控制台配置 CloudTrail 跟踪来记录 S3 存储桶的数据事件。

为 S3 通用存储桶或 S3 目录存储桶中的对象启用 CloudTrail 数据事件日志记录

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择存储桶的名称。
3. 选择 Properties (属性)。
4. 在 AWS CloudTrail data events (Amazon CloudTrail 数据事件) 下，请选择 Configure in CloudTrail (在 CloudTrail 中配置)。

   您可以创建新的 CloudTrail 跟踪或重复使用现有跟踪，并配置要在跟踪中记录的 Amazon S3 数据事件。有关如何在 CloudTrail 控制台中创建跟踪的信息，请参阅 AWS CloudTrail 用户指南中的使用控制台创建和更新跟踪。有关如何在 CloudTrail 控制台中配置 Amazon S3 数据事件日志记录的信息，请参阅 AWS CloudTrail 用户指南中的记录 Amazon S3 对象的数据事件。

   > **Note**
   >
   > 如果您使用 CloudTrail 控制台或 Amazon S3 控制台配置某个跟踪以记录 S3 存储桶的数据事件，Amazon S3 控制台将显示已为该存储桶启用对象级别日志记录。

为 S3 存储桶中的对象禁用 CloudTrail 数据事件日志记录

1. 登录到 AWS Management Console，然后通过以下网址打开 CloudTrail 控制台：https://console.aws.amazon.com/cloudtrail。
2. 在左侧导航窗格中，选择跟踪。
3. 选择您创建用于记录存储桶的事件的跟踪名称。
4. 在跟踪的详细信息页面上，选择右上角的停止记录。
5. 在随后显示的对话框中，选择停止记录。

有关创建 S3 存储桶时启用对象级别日志记录的信息，请参阅 创建存储桶。

有关使用 S3 存储桶进行 CloudTrail 日志记录的更多信息，请参阅以下主题：

- 查看 S3 存储桶的属性
- 使用 AWS CloudTrail 记录 Amazon S3 API 调用
- AWS CloudTrail 用户指南中的使用 CloudTrail 日志文件

使用 CloudTrail 识别 Amazon S3 请求

在 Amazon S3 中，可以使用 AWS CloudTrail 事件日志识别请求。AWS CloudTrail 是识别 Amazon S3 请求的首选方法，但是如果您使用的是 Amazon S3 服务器访问日志，请参阅the section called “识别 S3 请求”。

主题

- 识别 CloudTrail 日志中向 Amazon S3 发出的请求
- 使用 CloudTrail 识别 Amazon S3 签名版本 2 请求
- 使用 CloudTrail 识别对 S3 对象的访问权限

识别 CloudTrail 日志中向 Amazon S3 发出的请求

在设置 CloudTrail 以将事件传输到存储桶后，您应开始看到对象进入您在 Amazon S3 控制台上的目标存储桶。其格式如下所示：

```
s3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/Region/yyyy/mm/dd
```

由 CloudTrail 记录的事件作为使用 gzipped 进行压缩的 JSON 对象存储在您的 S3 存储桶中。要高效地查找请求，您应使用 Amazon Athena 等服务来为 CloudTrail 日志建立索引并查询。

有关 CloudTrail 和 Athena 的更多信息，请参阅《Amazon Athena 用户指南》中的使用分区投影在 Athena 中创建 AWS CloudTrail 日志表。

使用 CloudTrail 识别 Amazon S3 签名版本 2 请求

您可以使用 CloudTrail 事件日志来确定已用于在 Amazon S3 中签署请求的 API 签名版本。此功能非常重要，因为对 Signature Version 2 的支持将会关闭（弃用）。之后，Amazon S3 将不再接受使用 Signature Version 2 的请求，并且所有请求必须使用 Signature Version 4 进行签署。

我们强烈建议您使用 CloudTrail 来帮助确定您的任何工作流是否正在使用 Signature Version 2 进行签署。请通过将您的库和代码升级为使用 Signature Version 4 进行纠正，以防对您的业务产生任何影响。

有关更多信息，请参阅 AWS re:Post 中的公告：适用于 Amazon S3 的 AWS CloudTrail 为增强安全性审计添加了新字段。

> **Note**
>
> Amazon S3 的 CloudTrail 事件在请求详细信息中的键名称“additionalEventData”之下包含签名版本。要针对为 Amazon S3 中的对象发出的请求（如 GET、PUT 和 DELETE 请求）查找签名版本，您必须启用 CloudTrail 数据事件。（默认情况下，此特征处于关闭状态。）

AWS CloudTrail 是确定签名版本 2 请求的首选方法。如果您使用的是 Amazon S3 服务器访问日志，请参阅使用 Amazon S3 访问日志确定签名版本 2 请求。

主题

- 用于识别 Amazon S3 签名版本 2 请求的 Athena 查询示例
- 对签名版本 2 数据进行分区

用于识别 Amazon S3 签名版本 2 请求的 Athena 查询示例

Example — 选择所有签名版本 2 事件，并仅打印 **EventTime**、**S3_Action**、**Request_Parameters**、**Region**、**SourceIP** 和 **UserAgent**

在以下 Athena 查询中，将 *s3_cloudtrail_events_db.cloudtrail_table* 替换为您的 Athena 详细信息，并根据需要增加或删除限制。

```
SELECT EventTime, EventName as S3_Action, requestParameters as Request_Parameters,
 awsregion as AWS_Region, sourceipaddress as Source_IP, useragent as User_Agent
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
LIMIT 10;
```

Example — 选择发送签名版本 2 流量的所有请求者

```
SELECT useridentity.arn, Count(requestid) as RequestCount
```

```
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
    and json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
Group by useridentity.arn
```

对签名版本 2 数据进行分区

如果您有大量要查询的数据，则可以通过创建分区表来减少 Athena 的成本和运行时间。

为此，创建一个包含分区的新表，如下所示。

```
CREATE EXTERNAL TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned(
     eventversion STRING,
     userIdentity STRUCT<
         type:STRING,
         principalid:STRING,
         arn:STRING,
         accountid:STRING,
         invokedby:STRING,
         accesskeyid:STRING,
         userName:STRING,
      sessioncontext:STRUCT<
                 attributes:STRUCT<
                 mfaauthenticated:STRING,
                 creationdate:STRING>,
                 sessionIssuer:STRUCT<
                 type:STRING,
                 principalId:STRING,
                 arn:STRING,
                 accountId:STRING,
                 userName:STRING>
             >
          >,
     eventTime STRING,
     eventSource STRING,
     eventName STRING,
     awsRegion STRING,
     sourceIpAddress STRING,
     userAgent STRING,
     errorCode STRING,
     errorMessage STRING,
     requestParameters STRING,
```

```
        responseElements STRING,
        additionalEventData STRING,
        requestId STRING,
        eventId STRING,
        resources ARRAY<STRUCT<ARN:STRING,accountId: STRING,type:STRING>>,
        eventType STRING,
        apiVersion STRING,
        readOnly STRING,
        recipientAccountId STRING,
        serviceEventDetails STRING,
        sharedEventID STRING,
        vpcEndpointId STRING
    )
    PARTITIONED BY (region string, year string, month string, day string)
    ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
    STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
    LOCATION 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/';
```

然后，单独创建分区。您无法从尚未创建的日期获取结果。

```
ALTER TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned ADD
    PARTITION (region= 'us-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-east-1/2019/02/19/'
    PARTITION (region= 'us-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-west-1/2019/02/19/'
    PARTITION (region= 'us-west-2', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-west-2/2019/02/19/'
    PARTITION (region= 'ap-southeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-southeast-1/2019/02/19/'
    PARTITION (region= 'ap-southeast-2', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-southeast-2/2019/02/19/'
    PARTITION (region= 'ap-northeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-northeast-1/2019/02/19/'
    PARTITION (region= 'eu-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/eu-west-1/2019/02/19/'
    PARTITION (region= 'sa-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/sa-east-1/2019/02/19/';
```

然后，您可以根据这些分区发出请求，并且无需加载完整的存储桶。

```
SELECT useridentity.arn,
Count(requestid) AS RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_table_partitioned
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
AND region='us-east-1'
AND year='2019'
AND month='02'
AND day='19'
Group by useridentity.arn
```

使用 CloudTrail 识别对 S3 对象的访问权限

您可以使用 AWS CloudTrail 事件日志确定对于诸如 GetObject、DeleteObject 和 PutObject 等数据事件的 Amazon S3 对象访问请求，并发现有关这些请求的进一步信息。

以下示例说明如何从 AWS CloudTrail 事件日志获取对于 Amazon S3 的所有 PUT 对象请求。

主题

- 用于识别 Amazon S3 对象访问请求的 Athena 查询示例

用于识别 Amazon S3 对象访问请求的 Athena 查询示例

在以下 Athena 查询示例中，将 *s3_cloudtrail_events_db.cloudtrail_table* 替换为您的 Athena 详细信息，并根据需要修改日期范围。

Example — 选择所有具有 **PUT** 对象访问请求的事件，并仅打印 **EventTime**、**EventSource**、**SourceIP**、**UserAgent**、**BucketName**、**object** 和 **UserARN**

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
```

```
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'PutObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example — 选择所有具有 **GET** 对象访问请求的事件，并仅打印 **EventTime**、**EventSource**、**SourceIP**、**UserAgent**、**BucketName**、**object** 和 **UserARN**

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'GetObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example — 选择特定时段内向存储桶发送的所有匿名请求者事件，并仅打印 **EventTime**、**EventName**、**EventSource**、**SourceIP**、**UserAgent**、**BucketName**、**UserARN** 和 **AccountID**

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  userIdentity.arn as userArn,
  userIdentity.accountId
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  userIdentity.accountId = 'anonymous'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example — 识别所有需要 ACL 进行授权的请求

以下 Amazon Athena 查询示例说明如何识别向 S3 存储桶发出的所有请求，这些请求需要访问控制列表（ACL）进行授权。如果请求需要 ACL 进行授权，则 additionalEventData 中的 aclRequired 值为 Yes。如果不需要 ACL，则 aclRequired 不存在。您可以使用此信息将这些 ACL 权限迁移到相应的存储桶策略。创建这些存储桶策略后，您可以对这些存储桶禁用 ACL。有关禁用 ACL 的更多信息，请参阅禁用 ACL 的先决条件。

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  userIdentity.arn as userArn,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  json_extract_scalar(additionalEventData, '$.aclRequired') as aclRequired
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  json_extract_scalar(additionalEventData, '$.aclRequired') = 'Yes'
  AND eventTime BETWEEN '2022-05-10T00:00:00Z' and '2022-08-10T00:00:00Z'
```

Note

- 也可以使用这些查询示例进行安全监控。您可以查看意外或未经授权的 IP 地址或请求者发出的 PutObject 或 GetObject 调用的结果，以及确定向存储桶发出的任何匿名请求。
- 此查询仅从启用了日志记录的时间检索信息。

如果您使用的是 Amazon S3 服务器访问日志，请参阅 使用 Amazon S3 访问日志确定对象访问请求。

使用服务器访问日志记录来记录请求

服务器访问日志记录详细地记录对存储桶提出的各种请求。对于许多应用程序而言，服务器访问日志很有用。例如，访问日志信息可能在安全和访问权限审核方面很有用。此信息还有助于您了解客户群以及您的 Amazon S3 账单。

> **Note**
>
> 服务器访问日志不会记录 2019 年 3 月 20 日之后发布的区域的不正确区域重定向错误的相关信息。在存储桶所在区域的外部发出对象或存储桶的请求时，会发生不正确区域重定向错误。

如何启用日志传送？

要启用日志传输，请执行以下基本步骤。有关更多信息，请参阅 启用 Amazon S3 服务器访问日志记录。

1. 提供目的地存储桶（也称为目标存储桶）的名称。此存储桶是您希望 Amazon S3 用于将访问日志保存为对象的位置。源存储桶和目标存储桶必须位于同一个 AWS 区域，并且由同一个账户拥有。目标存储桶不得具有 S3 对象锁定默认保留期配置。目标存储桶也不得启用申请方付款。

   您可以让日志传输至您拥有的且与源存储桶位于同一区域中的任何存储桶，包括源存储桶本身。不过，为了更方便地管理日志，我们建议您将访问日志保存在不同的存储桶中。

   当源存储桶和目标存储桶是同一存储桶时，将为写入该存储桶的日志创建额外的日志，从而形成日志的无限循环。我们不建议这样做，因为它会导致您的存储账单金额小幅增加。此外，有关日志的额外日志可能会导致更难以找到您所查找的日志。

   如果您选择将访问日志保存在源存储桶中，建议您为所有日志对象键指定目的地前缀（也称为目标前缀）。指定前缀时，所有日志对象名称都以公共字符串开头，从而使日志对象更易识别。

2. （可选）将一个目标前缀分配给所有 Amazon S3 日志对象键。利用目的地前缀（也称作目标前缀），可更轻松地找到日志对象。例如，如果您指定前缀值 `logs/`，则 Amazon S3 创建的每个日志对象的键均以 `logs/` 前缀开头，例如：

   ```
   logs/2013-11-01-21-32-16-E568B2907131C0C0
   ```

   如果指定前缀值 `logs`，则日志对象如下所示：

   ```
   logs2013-11-01-21-32-16-E568B2907131C0C0
   ```

   当同一目标存储桶有多个存储桶日志时，键前缀也可用于区分源存储桶。

   当您删除日志时，此前缀也很有用。例如，您可以设置一个生命周期配置规则，让 Amazon S3 删除具有特定前缀的对象。有关更多信息，请参阅 删除 Amazon S3 日志文件。

3. （可选）设置权限以使其他人能够访问生成的日志。默认情况下，仅存储桶拥有者始终拥有日志对象的完全访问权限。如果您的目标存储桶使用 S3 对象所有权的强制存储桶拥有者设置来禁用访问控制列表（ACL），则无法在使用 ACL 的目的地授权（也称作目标授权）中授予权限。但是，您可以更新目标存储桶的存储桶策略以向其他存储桶授予访问权限。有关更多信息，请参阅Amazon S3 的身份和访问管理 和日志传输的权限。
4. （可选）为日志文件设置日志对象键格式。对于日志对象键格式（也称为目标对象键格式），有两个选项：
   - 非基于日期的分区 - 这是原始日志对象键格式。如果选择此格式，则日志文件键格式如下所示：

     ```
     [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
     ```

     例如，如果您指定 `logs/` 作为前缀，则日志对象将按以下方式命名：

     ```
     logs/2013-11-01-21-32-16-E568B2907131C0C0
     ```

   - 基于日期的分区 - 如果选择基于日期的分区，则可以选择日志文件的事件时间或传输时间作为日志格式中使用的日期源。此格式让查询日志变得更轻松。

     如果选择基于日期的分区，则日志文件键格式如下所示：

     ```
     [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/
     [DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
     ```

     例如，如果您指定 `logs/` 作为目标前缀，则日志对象将按以下方式命名：

     ```
     logs/123456789012/us-west-2/DOC-EXAMPLE-SOURCE-
     BUCKET/2023/03/01/2023-03-01-21-32-16-E568B2907131C0C0
     ```

     对于传输时间传送，日志文件名中的时间与日志文件的传输时间对应。

     对于事件时间传送，年、月和日对应于事件的发生日期，小时、分钟和秒设置为键中的 `00`。这些日志文件中传送的日志仅针对特定日期。

如果您通过 AWS Command Line Interface（AWS CLI）、AWS SDK 或 Amazon S3 REST API 配置日志，请使用 `TargetObjectKeyFormat` 指定日志对象键格式。要指定非基于日期的分区，请使用 `SimplePrefix`。要指定基于日期的分区，请使用 `PartitionedPrefix`。

如果使用 PartitionedPrefix，则使用 PartitionDateSource 来指定 EventTime 或 DeliveryTime。

对于 SimplePrefix，日志文件键格式如下所示：

```
[TargetPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

对于带事件时间或传输时间的 PartitionedPrefix，日志文件键格式如下所示：

```
[TargetPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

日志对象密钥格式

Amazon S3 将以下对象键格式用于在目标存储桶中上传的日志对象：

- 非基于日期的分区 - 这是原始日志对象键格式。如果选择此格式，则日志文件键格式如下所示：

```
[DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

- 基于日期的分区 - 如果选择基于日期的分区，则可以选择日志文件的事件时间或传输时间作为日志格式中使用的日期源。此格式让查询日志变得更轻松。

  如果选择基于日期的分区，则日志文件键格式如下所示：

```
[DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/
[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]
```

在日志对象键中，YYYY、MM、DD、hh、mm 和 ss 分别是表示年、月、日、小时、分钟和秒的数字。这些日期和时间采用协调世界时 (UTC)。

在特定时间传输的日志文件可包含在该时间前的任何时刻编写的记录。无法知道是否已传输特定时间间隔内的所有日志记录。

键的 UniqueString 部分用于防止覆盖文件。它没有意义，日志处理软件应忽略它。

如何传输日志？

Amazon S3 定期收集访问日志记录，在日志文件中整合这些记录，然后将日志文件作为日志对象上传到目标存储桶。如果多个启用了日志记录的源存储桶使用相同的目标存储桶，此目标存储桶中将保留所有这些源存储桶的访问日志。但是，每个日志对象只会报告特定源存储桶的访问日志记录。

Amazon S3 使用特殊的日志传输账户写入服务器访问日志。这些写入受常规的访问控制限制。我们建议您更新目标存储桶上的存储桶策略，以授予对日志记录服务主体（`logging.s3.amazonaws.com`）的访问权限来进行访问日志传输。您也可以通过存储桶访问控制列表（ACL）向 S3 日志传输组授予访问日志传输的访问权限。但是，建议不要使用存储桶 ACL 授予对 S3 日志传输组的访问权限。

当您启用服务器访问日志记录并通过目标存储桶策略授予访问日志传输的访问权限时，您必须更新该策略，以允许日志记录服务主体访问 `s3:PutObject`。如果您使用 Amazon S3 控制台启用服务器访问日志录入，该控制台会自动更新目标存储桶策略，以便将这些权限授予日志记录服务主体。有关授予服务器访问日志传输的权限的更多信息，请参阅 日志传输的权限。

> **Note**
>
> 对于 VPC 端点策略拒绝的 VPC 端点请求，或者对于在评估 VPC 策略之前失败的请求，S3 不支持向请求者或存储桶拥有者传输 CloudTrail 日志或服务器访问日志。

S3 对象所有权的强制存储桶拥有者设置

如果目标存储桶使用对象所有权的强制存储桶拥有者设置，ACL 将被禁用，并且不再影响权限。您必须更新目标存储桶上的存储桶策略，以授予对日志记录服务主体的访问权。有关对象所有权的更多信息，请参阅为服务器访问日志记录授予对 S3 日志传输组的访问权限。

最大努力服务器日志传输

服务器访问日志记录会以最大努力进行传输。针对已正确配置了日志记录的存储桶的大多数请求会导致传输一条日志记录。大多数日志记录将在记录后的几小时内传输，但可以更频繁地传输这些记录。

因此不能保证服务器日志记录的完整性和即时性。特殊请求的日志记录可能会在实际处理了请求之后进行传输，也可能根本不会传输。您甚至可能会看到日志记录的副本。服务器日志的用途在于向您提供有关存储桶流量性质方面的信息。虽然日志记录丢失或重复的情况十分少见，但请注意，服务器日志记录并不旨在完整记录所有请求。

由于服务器日志记录特征的最大努力性质，使用情况报告中可能有一个或多个访问请求不会出现在传输的服务器日志中。您可以在 AWS Billing and Cost Management 控制台的成本和使用情况报告下找到这些使用情况报告。

存储桶日志记录状态更改将逐渐生效

存储桶日志记录状态的更改需要一定时间才能实际影响日志文件的传输。例如，如果您为某个存储桶启用了日志记录，那么将记录在以下时间内发送的请求，而不会记录其他请求。假定您将日志记录的目标存储桶从存储桶 A 更改为存储桶 B，则在接下来的一个小时里仍可能有一些日志传输到存储桶 A，但其他日志则会传输到新的目标存储桶 B。无论如何，新的设置将最终生效，并且您无需执行任何操作。

有关日志记录和日志文件的更多信息，请参阅以下各部分：

主题

- 启用 Amazon S3 服务器访问日志记录
- Amazon S3 服务器访问日志格式
- 删除 Amazon S3 日志文件
- 使用 Amazon S3 服务器访问日志来确定请求

启用 Amazon S3 服务器访问日志记录

服务器访问日志记录详细地记录对 Amazon S3 存储桶提出的各种请求。对于许多应用程序而言，服务器访问日志很有用。例如，访问日志信息可能在安全和访问权限审核方面很有用。此信息还有助于您了解客户群以及您的 Amazon S3 账单。

默认情况下，Amazon S3 不会收集服务器访问日志。在您启用日志记录后，Amazon S3 会将源存储桶的访问日志传输到您选择的目的地存储桶（也称作目标存储桶）。目标存储桶必须位于源存储桶所在的 AWS 区域和 AWS 账户中。

访问日志记录包含有关对存储桶做出的请求的详细信息。这些信息可能包括请求类型、请求中指定的资源以及处理请求的时间和日期。有关日志记录基本知识的更多信息，请参阅 使用服务器访问日志记录来记录请求。

> **Important**
>
> - 在 Amazon S3 存储桶上启用服务器访问日志记录不收取额外费用。但是，系统提交给您的任何日志文件都会产生普通存储费用。(您可以随时删除日志文件。) 我们不会估计传输日志文件的数据传输费，但会按正常数据传输费率对访问日志文件收费。

- 您的目标存储桶不应启用服务器访问日志记录。您可以让日志传输至您拥有的且与源存储桶位于同一区域中的任何存储桶，包括源存储桶本身。但是，将日志传输到源存储桶会导致日志的无限循环，因此不建议这样做。为了更方便地管理日志，我们建议您将访问日志保存在不同的存储桶中。有关更多信息，请参阅 如何启用日志传送？
- 已启用 S3 对象锁定的 S3 存储桶不能用作服务器访问日志的目标存储桶。目标存储桶不得具有默认保留期配置。
- 目标存储桶不得启用“申请方付款”。
- 仅当使用具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）（该加密方式使用 256 位高级加密标准（AES-256））时，才能对目标存储桶使用默认存储桶加密。不支持默认的具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）。

您可以使用 Amazon S3 控制台、Amazon S3 API、AWS Command Line Interface（AWS CLI）或 AWS SDK 启用或禁用服务器访问日志记录。

日志传输的权限

Amazon S3 使用特殊的日志传输账户写入服务器访问日志。这些写入受常规的访问控制限制。对于访问日志传输，您必须向日志记录服务主体（`logging.s3.amazonaws.com`）授予对目标存储桶的访问权限。

要向 Amazon S3 授予日志传输权限，您可以使用存储桶策略或存储桶访问控制列表（ACL），具体取决于目标存储桶的 S3 对象所有权设置。但是，建议您使用存储桶策略，而不是 ACL。

S3 对象所有权的强制存储桶拥有者设置

如果目标存储桶使用对象所有权的强制存储桶拥有者设置，ACL 将被禁用，并且不再影响权限。在这种情况下，您必须更新目标存储桶的存储桶策略，以向日志记录服务主体授予访问权限。您无法更新存储桶 ACL 以授予对 S3 日志传输组的访问权限。您也无法在 PutBucketLogging 配置中包含目的地授权（也称为目标授权）。

有关将现有存储桶 ACL 以便将访问日志传输迁移到存储桶策略的信息，请参阅 为服务器访问日志记录授予对 S3 日志传输组的访问权限。有关对象所有权的更多信息，请参阅为您的存储桶控制对象所有权和禁用 ACL。。当您创建新存储桶时，默认情况下 ACL 处于禁用状态。

使用存储桶策略授予访问权限

要使用目标存储桶的存储桶策略授予访问权限，请更新存储桶策略以向日志记录服务主体授予 `s3:PutObject` 权限。如果您使用 Amazon S3 控制台启用服务器访问日志记录，该控制台会自动更

新目标存储桶的存储桶策略，以便将此权限授予日志记录服务主体。如果以编程方式启用服务器访问日志记录，则必须手动更新目标存储桶的存储桶策略以向日志记录服务主体授予访问权限。

有关向日志服务主体授予访问权限的存储桶策略示例，请参阅the section called “使用存储桶策略向日志记录服务主体授予权限”。

使用存储桶 ACL 授予访问权限

您也可以使用存储桶 ACL 授予访问日志传输的访问权限。您可以在存储桶 ACL 中添加一个授予条目，以授予对 S3 日志传输组的 WRITE 和 READ_ACP 权限。但是，不建议使用存储桶 ACL 授予对 S3 日志传输组的访问权限。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。有关将现有存储桶 ACL 以便将访问日志传输迁移到存储桶策略的信息，请参阅 为服务器访问日志记录授予对 S3 日志传输组的访问权限。有关向日志记录服务主体授予访问权限的示例 ACL，请参阅the section called “使用存储桶 ACL 向日志传输组授予权限”。

使用存储桶策略向日志记录服务主体授予权限

此示例存储桶策略向日志记录服务主体（logging.s3.amazonaws.com）授予 s3:PutObject 权限。要使用这一存储桶策略，请将 *user input placeholders* 替换为您自己的信息。在以下策略中，*amzn-s3-demo-destination-bucket* 是将服务器访问日志传输到的目标存储桶，*amzn-s3-demo-source-bucket* 是源存储桶。*EXAMPLE-LOGGING-PREFIX* 是要用于日志对象的可选目的地前缀（也称为目标前缀）。*SOURCE-ACCOUNT-ID* 是拥有源存储桶的 AWS 账户。

> **Note**
>
> 如果存储桶策略中有 Deny 语句，请确保这些语句不会阻止 Amazon S3 传输访问日志。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/EXAMPLE-LOGGING-
PREFIX*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

使用存储桶 ACL 向日志传输组授予权限

> **Note**
>
> 作为安全最佳实践，默认情况下，Amazon S3 在所有新存储桶中禁用访问控制列表（ACL）。有关使用 Amazon S3 控制台 ACL 权限的更多信息，请参阅 配置 ACL。

我们建议您不要使用此方法，而是使用存储桶 ACL 向日志传输组授予权限。但是，如果目标存储桶使用对象所有权的强制存储桶拥有者设置，则无法设置存储桶或对象 ACL。您也无法在 PutBucketLogging 配置中包含目的地授权（也称为目标授权）。而是必须使用存储桶策略向日志记录服务主体（`logging.s3.amazonaws.com`）授予访问权限。有关更多信息，请参阅 日志传输的权限。

在存储桶 ACL 中，日志传输组通过以下 URL 表示：

```
http://acs.amazonaws.com/groups/s3/LogDelivery
```

要授予 `WRITE` 和 `READ_ACP`（ACL 读取）权限，请将以下授权添加到目标存储桶 ACL：

```
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>WRITE</Permission>
</Grant>
```

```
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>READ_ACP</Permission>
</Grant>
```

有关以编程方式添加 ACL 授权的示例，请参阅 配置 ACL。

> **Important**
>
> 当您对存储桶使用 AWS CloudFormation 来启用 Amazon S3 服务器访问日志记录，并使用 ACL 向 S3 日志传输组授予访问权限时，您还必须向 CloudFormation 模板添加 `"AccessControl": "LogDeliveryWrite"`。这样做非常重要，因为您只能通过为存储桶创建 ACL 来授予这些权限，而无法在 CloudFormation 中为存储桶创建自定义 ACL。您只能将标准 ACL 与 CloudFormation 结合使用。

要启用服务器访问日志记录

要使用 Amazon S3 控制台、Amazon S3 REST API、AWS SDK 和 AWS CLI 启用服务器访问日志记录，请使用以下过程。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要为其启用服务器访问日志记录的存储桶的名称。
3. 选择 Properties (属性)。
4. 在 Server access logging (服务器访问日志记录) 部分中，请选择 Edit (编辑)。
5. 在服务器访问日志记录下，选择启用。
6. 在目标存储桶下，指定存储桶和可选前缀。如果您指定前缀，建议您在前缀后面加上正斜杠（/），以便更轻松地查找日志。

> **Note**
>
> 通过斜杠（/）指定前缀可让您更轻松地查找日志对象。例如，如果您指定前缀值 `logs/`，则 Amazon S3 创建的每个日志对象的键均以 `logs/` 前缀开头，如下所示：

```
logs/2013-11-01-21-32-16-E568B2907131C0C0
```

如果指定前缀值 `logs`，则日志对象如下所示：

```
logs2013-11-01-21-32-16-E568B2907131C0C0
```

7. 在日志对象件格式下，执行下列操作之一：

   - 要选择非基于日期的分区，请选择 [DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]。
   - 要选择基于日期的分区，请选择 [DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]，然后选择 S3 事件时间或日志文件传输时间。

8. 选择 Save changes（保存更改）。

   当您在存储桶上启用服务器访问日志记录时，控制台会在源存储桶上启用日志记录并更新目标存储桶的存储桶策略，以便向日志记录服务主体（`logging.s3.amazonaws.com`）授予 `s3:PutObject` 权限。有关存储桶策略的更多信息，请参阅 使用存储桶策略向日志记录服务主体授予权限。

   您可以查看目标存储桶中的日志。启用服务器访问日志记录后，可能需要数小时，日志才会传输到目标存储桶。有关如何以及何时传输日志的更多信息，请参阅如何传输日志？。

有关更多信息，请参阅 查看 S3 存储桶的属性。

使用 REST API

要启用日志记录，请提交 PutBucketLogging 请求，以在源存储桶上添加日志记录配置。该请求指定目的地存储桶（也称作目标存储桶），并（可选）指定要用于所有日志对象键的前缀。

以下示例将 *amzn-s3-demo-destination-bucket* 标识为目标存储桶，将 *logs/* 标识为前缀。

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>amzn-s3-demo-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

以下示例将 *amzn-s3-demo-destination-bucket* 标识为目标存储桶，将 *logs/* 标识为前缀，并将 EventTime 标识为日志对象键格式。

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>amzn-s3-demo-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
    <TargetObjectKeyFormat>
      <PartitionedPrefix>
         <PartitionDateSource>EventTime</PartitionDateSource>
      </PartitionedPrefix>
  </TargetObjectKeyFormat>
  </LoggingEnabled>
</BucketLoggingStatus>
```

日志对象由 S3 日志传递账户编写并拥有，存储桶拥有者对日志对象授予完全权限。您可以选择性地使用目的地授权（也称作目标授权）向其他用户授予权限，以便他们能够访问日志。有关更多信息，请参阅 PutBucketLogging。

> **Note**
>
> 如果目标存储桶使用对象所有权的强制存储桶拥有者设置，则您无法使用目标授权向其他用户授予权限。要向其他用户授予权限，您可以更新目标存储桶的存储桶策略。有关更多信息，请参阅 日志传输的权限。

要检索存储桶上的日志记录配置，请使用 GetBucketLogging API 操作。

要删除日志记录配置，请发送带空白 BucketLoggingStatus 的 PutBucketLogging 请求：

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
</BucketLoggingStatus>
```

要对存储桶启用日志记录，您可以使用 Amazon S3 API 或 AWS SDK 包装程序库。

使用 AWS SDK

以下示例对存储桶启用日志记录。您必须创建两个存储桶，即一个源存储桶和一个目的地（目标）存储桶。这些示例首先更新目标存储桶上的存储桶 ACL。之后，向日志传输组授予向目标存储桶写入日志所需的权限，然后在源存储桶上启用日志记录。

这些示例不适用于使用对象所有权的强制存储桶拥有者设置的目标存储桶。

如果目的地（目标）存储桶使用对象所有权的强制存储桶拥有者设置，则无法设置存储桶或对象 ACL。您也不能将目的地（目标）授权包含在 PutBucketLogging 配置中。您必须使用存储桶策略向日志服务主体（`logging.s3.amazonaws.com`）授予访问权限。有关更多信息，请参阅 日志传输的权限。

.NET

AWS SDK for .NET

> **Note**
>
> 在 GitHub 上查看更多内容。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
{
    private static IConfiguration _configuration = null!;

    public static async Task Main()
    {
        LoadConfig();

        string bucketName = _configuration["BucketName"];
        string logBucketName = _configuration["LogBucketName"];
        string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
```

```
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
```

```
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
```

```
        /// This method enables logging for an Amazon S3 bucket. Logs will be
 stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
```

```
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
  local settings.
                .Build();
        }
    }
```

- 有关更多信息，请参阅《AWS SDK for .NET API 参考》中的 PutBucketLogging。

Java

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLoggingStatus;
import software.amazon.awssdk.services.s3.model.LoggingEnabled;
import software.amazon.awssdk.services.s3.model.PartitionedPrefix;
import software.amazon.awssdk.services.s3.model.PutBucketLoggingRequest;
import software.amazon.awssdk.services.s3.model.TargetObjectKeyFormat;

// Class to set a bucket policy on a target S3 bucket and enable server access
 logging on a source S3 bucket.
public class ServerAccessLogging {
    private static S3Client s3Client;

    public static void main(String[] args) {
        String sourceBucketName = "SOURCE-BUCKET";
        String targetBucketName = "TARGET-BUCKET";
        String sourceAccountId = "123456789012";
        String targetPrefix = "logs/";

        // Create S3 Client.
        s3Client = S3Client.builder().
                region(Region.US_EAST_2)
                .build();

        // Set a bucket policy on the target S3 bucket to enable server access
 logging by granting the
```

```
        // logging.s3.amazonaws.com principal permission to use the PutObject
 operation.
        ServerAccessLogging serverAccessLogging = new ServerAccessLogging();
        serverAccessLogging.setTargetBucketPolicy(sourceAccountId, sourceBucketName,
 targetBucketName);

        // Enable server access logging on the source S3 bucket.
        serverAccessLogging.enableServerAccessLogging(sourceBucketName,
 targetBucketName,
                targetPrefix);

    }

    // Function to set a bucket policy on the target S3 bucket to enable server
 access logging by granting the
    // logging.s3.amazonaws.com principal permission to use the PutObject operation.
    public void setTargetBucketPolicy(String sourceAccountId, String
 sourceBucketName, String targetBucketName) {
        String policy = "{\n" +
                "    \"Version\": \"2012-10-17\",\n" +
                "    \"Statement\": [\n" +
                "        {\n" +
                "            \"Sid\": \"S3ServerAccessLogsPolicy\",\n" +
                "            \"Effect\": \"Allow\",\n" +
                "            \"Principal\": {\"Service\": \"logging.s3.amazonaws.com
\"},\n" +
                "            \"Action\": [\n" +
                "                \"s3:PutObject\"\n" +
                "            ],\n" +
                "            \"Resource\": \"arn:aws:s3:::" + targetBucketName + "/*
\",\n" +
                "            \"Condition\": {\n" +
                "                \"ArnLike\": {\n" +
                "                    \"aws:SourceArn\": \"arn:aws:s3:::" +
 sourceBucketName + "\"\n" +
                "                },\n" +
                "                \"StringEquals\": {\n" +
                "                    \"aws:SourceAccount\": \"" + sourceAccountId +
 "\"\n" +
                "                }\n" +
                "            }\n" +
                "        }\n" +
                "    ]\n" +
                "}";
```

```
        s3Client.putBucketPolicy(b -> b.bucket(targetBucketName).policy(policy));
    }

    // Function to enable server access logging on the source S3 bucket.
    public void enableServerAccessLogging(String sourceBucketName, String
targetBucketName,
            String targetPrefix) {
        TargetObjectKeyFormat targetObjectKeyFormat =
TargetObjectKeyFormat.builder()

.partitionedPrefix(PartitionedPrefix.builder().partitionDateSource("EventTime").build())
                .build();
        LoggingEnabled loggingEnabled = LoggingEnabled.builder()
                .targetBucket(targetBucketName)
                .targetPrefix(targetPrefix)
                .targetObjectKeyFormat(targetObjectKeyFormat)
                .build();
        BucketLoggingStatus bucketLoggingStatus = BucketLoggingStatus.builder()
                .loggingEnabled(loggingEnabled)
                .build();
        s3Client.putBucketLogging(PutBucketLoggingRequest.builder()
                .bucket(sourceBucketName)
                .bucketLoggingStatus(bucketLoggingStatus)
                .build());
    }

}
```

使用 AWS CLI

我们建议您在其中放置 S3 存储桶的每个 AWS 区域 中都创建一个专用日志记录存储桶。然后，将 Amazon S3 访问日志传输到该 S3 存储桶。有关更多信息和示例，请参阅《AWS CLI 参考》中的 put-bucket-logging。

如果目的地（目标）存储桶使用对象所有权的强制存储桶拥有者设置，则无法设置存储桶或对象 ACL。您也不能将目的地（目标）授权包含在 PutBucketLogging 配置中。您必须使用存储桶策略向日志服务主体（`logging.s3.amazonaws.com`）授予访问权限。有关更多信息，请参阅 日志传输的权限。

Example — 为跨两个区域的五个存储桶启用访问日志

在本示例中，您有以下 5 个存储桶：

- 1-amzn-s3-demo-bucket1-us-east-1
- 2-amzn-s3-demo-bucket1-us-east-1
- 3-amzn-s3-demo-bucket1-us-east-1
- 1-amzn-s3-demo-bucket1-us-west-2
- 2-amzn-s3-demo-bucket1-us-west-2

> **Note**
>
> 以下过程的最后一步提供了示例 bash 脚本，可使用这些脚本来创建日志记录存储桶，并在这些存储桶上启用服务器访问日志记录。要使用这些脚本，必须创建 policy.json 和 logging.json 文件，如以下过程所述。

1. 在美国西部（俄勒冈州）和美国东部（弗吉尼亚州北部）区域创建两个日志记录存储桶，并为其指定以下名称：
   - amzn-s3-demo-bucket1-logs-us-east-1
   - amzn-s3-demo-bucket1-logs-us-west-2
2. 稍后，在这些步骤中，您将启用服务器访问日志记录，如下所示：
   - 1-amzn-s3-demo-bucket1-us-east-1 记录到带有前缀 1-amzn-s3-demo-bucket1-us-east-1 的 S3 存储桶 amzn-s3-demo-bucket1-logs-us-east-1
   - 2-amzn-s3-demo-bucket1-us-east-1 记录到带有前缀 2-amzn-s3-demo-bucket1-us-east-1 的 S3 存储桶 amzn-s3-demo-bucket1-logs-us-east-1
   - 3-amzn-s3-demo-bucket1-us-east-1 记录到带有前缀 3-amzn-s3-demo-bucket1-us-east-1 的 S3 存储桶 amzn-s3-demo-bucket1-logs-us-east-1
   - 1-amzn-s3-demo-bucket1-us-west-2 记录到带有前缀 1-amzn-s3-demo-bucket1-us-west-2 的 S3 存储桶 amzn-s3-demo-bucket1-logs-us-west-2
   - 2-amzn-s3-demo-bucket1-us-west-2 记录到带有前缀 2-amzn-s3-demo-bucket1-us-west-2 的 S3 存储桶 amzn-s3-demo-bucket1-logs-us-west-2
3. 对于每个目标日志记录存储桶，使用存储桶 ACL 或存储桶策略授予服务器访问日志传输的权限：
   - 更新存储桶策略（建议）– 要向日志记录服务主体授予权限，请使用以下 put-bucket-policy 命令：将 *amzn-s3-demo-destination-bucket-logs* 替换为您的目标存储桶的名称。

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-destination-bucket-logs --
policy file://policy.json
```

Policy.json 是当前文件夹中包含以下存储桶策略的 JSON 文档。要使用这一存储桶策略，请将 *user input placeholders* 替换为您自己的信息。在以下策略中，*amzn-s3-demo-destination-bucket-logs* 是要将服务器访问日志传输到的目标存储桶，*amzn-s3-demo-source-bucket* 是源存储桶。*SOURCE-ACCOUNT-ID* 是拥有源存储桶的 AWS 账户。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket-logs/*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

- 更新存储桶 ACL –要向 S3 日志传输组授予权限，请使用以下 put-bucket-acl 命令。将 *amzn-s3-demo-destination-bucket-logs* 替换为您的目的地（目标）存储桶的名称。

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-destination-bucket-logs  --
grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp
 URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

4. 然后，创建一个包含日志记录配置的 logging.json 文件（基于以下三个示例之一）。创建 logging.json 文件后，可以使用以下 put-bucket-logging 命令应用日志记录配置。将 *amzn-s3-demo-destination-bucket-logs* 替换为您的目的地（目标）存储桶的名称。

```
aws s3api put-bucket-logging --bucket amzn-s3-demo-destination-bucket-logs --
bucket-logging-status file://logging.json
```

> **Note**
>
> 您可以使用下一步中提供的 bash 脚本之一，而不是使用此 put-bucket-logging 命令在每个目标存储桶上应用日志记录配置。要使用这些脚本，您必须创建 policy.json 和 logging.json 文件，如该过程所述。

logging.json 文件是当前文件夹中包含日志记录配置的 JSON 文档。如果目标存储桶使用对象所有权的强制存储桶拥有者设置，则您的日志记录配置不能包含目的地（目标）授权。有关更多信息，请参阅 日志传输的权限。

Example – **logging.json**，不带目的地（目标）授权

以下示例 logging.json 文件不包含目的地（目标）授权。因此，您可以将此配置应用于使用对象所有权的强制存储桶拥有者设置的目的地（目标）存储桶。

```
{
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/"
     }
 }
```

Example – **logging.json**，带目的地（目标）授权

以下示例 logging.json 文件包含目的地（目标）授权。

如果目标存储桶使用对象所有权的强制存储桶拥有者设置，则您无法在 PutBucketLogging 配置中包含目的地（目标）授权。有关更多信息，请参阅 日志传输的权限。

```
{
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/",
        "TargetGrants": [
             {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "user@example.com"
                 },
                "Permission": "FULL_CONTROL"
             }
         ]
     }
 }
```

Example – **logging.json**，日志对象键格式设置为 S3 事件时间

以下 logging.json 文件将日志对象键格式更改为 S3 事件时间。有关设置日志对象键格式的更多信息，请参阅the section called “如何启用日志传送？”

```
{
  "LoggingEnabled": {
      "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
      "TargetPrefix": "amzn-s3-demo-destination-bucket/",
      "TargetObjectKeyFormat": {
          "PartitionedPrefix": {
              "PartitionDateSource": "EventTime"
          }
       }
  }
```

```
}
```

5. 使用下列 bash 脚本之一为您账户中的所有存储桶添加访问日志记录。将 *amzn-s3-demo-destination-bucket-logs* 替换为目的地（目标）存储桶的名称，并将 *us-west-2* 替换为存储桶所在的区域的名称。

> **Note**
>
> 此脚本仅在所有存储桶均位于同一区域中时有用。如果您在多个区域中有存储桶，则必须调整该脚本。

Example — 使用存储桶策略授予访问权限并为账户中的存储桶添加日志记录

```
loggingBucket='amzn-s3-demo-destination-bucket-logs'
region='us-west-2'


# Create the logging bucket.
aws s3 mb s3://$loggingBucket --region $region

aws s3api put-bucket-policy --bucket $loggingBucket --policy file://policy.json

# List the buckets in this account.
buckets="$(aws s3 ls | awk '{print $3}')"

# Put a bucket logging configuration on each bucket.
for bucket in $buckets
    do
      # This if statement excludes the logging bucket.
      if [ "$bucket" != "$loggingBucket" ] ; then
          continue;
      fi
      printf '{
        "LoggingEnabled": {
          "TargetBucket": "%s",
          "TargetPrefix": "%s/"
      }
    }' "$loggingBucket" "$bucket"  > logging.json
```

```
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
      echo "$bucket done"
  done

  rm logging.json

  echo "Complete"
```

Example — 使用存储桶 ACL 授予访问权限并为账户中的存储桶添加日志记录

```
  loggingBucket='amzn-s3-demo-destination-bucket-logs'
  region='us-west-2'


  # Create the logging bucket.
  aws s3 mb s3://$loggingBucket --region $region

  aws s3api put-bucket-acl --bucket $loggingBucket --grant-write URI=http://
acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp URI=http://
acs.amazonaws.com/groups/s3/LogDelivery

  # List the buckets in this account.
  buckets="$(aws s3 ls | awk '{print $3}')"

  # Put a bucket logging configuration on each bucket.
  for bucket in $buckets
      do
        # This if statement excludes the logging bucket.
        if [ "$bucket" != "$loggingBucket" ] ; then
            continue;
        fi
        printf '{
          "LoggingEnabled": {
            "TargetBucket": "%s",
            "TargetPrefix": "%s/"
        }
      }' "$loggingBucket" "$bucket"  > logging.json
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
      echo "$bucket done"
  done
```

```
rm logging.json

echo "Complete"
```

验证服务器访问日志设置

启用服务器访问日志记录后，请完成以下步骤：

- 访问目标存储桶并验证是否正在传输日志文件。设置访问日志后，Amazon S3 会立即开始捕获请求并记录这些请求。但是，日志会在数小时后传输到目标存储桶。有关更多信息，请参阅the section called “存储桶日志记录状态更改将逐渐生效” 和the section called “最大努力服务器日志传输”。

  还可以通过使用 Amazon S3 请求指标并为这些指标设置 Amazon CloudWatch 警报，来自动验证日志传输。有关更多信息，请参阅 使用 Amazon CloudWatch 监控指标。
- 验证您是否能够打开和读取日志文件的内容。

有关服务器访问日志记录故障排除信息，请参阅排查服务器访问日志记录问题。

Amazon S3 服务器访问日志格式

服务器访问日志记录详细地记录对 Amazon S3 存储桶提出的各种请求。您可以将服务器访问日志用于以下目的：

- 进行安全和访问审计
- 了解您的客户群
- 了解您的 Amazon S3 账单

本节介绍了有关 Amazon S3 服务器访问日志文件的格式和其他详细信息。

服务器访问日志文件由一系列的换行分隔日志记录组成。每个日志记录表示一个请求并由空格分隔的字段组成。

以下是含有五份日志记录的示例日志。

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F3EXAMPLE
```

```
 REST.GET.VERSIONING - "GET /amzn-s3-demo-bucket1?versioning HTTP/1.1" 200 - 113 - 7 -
 "-" "S3Console/0.4" - s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/
XV/VLi31234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-demo-bucket1.s3.us-
west-1.amazonaws.com TLSV1.2 arn:aws:s3:us-west-1:123456789012:accesspoint/example-AP
 Yes
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 891CE47D2EXAMPLE
 REST.GET.LOGGING_STATUS - "GET /amzn-s3-demo-bucket1?logging HTTP/1.1" 200 -
 242 - 11 - "-" "S3Console/0.4" - 9vKBE6vMhrNiWHZmb2L0mXOcqPGzQOI5XLnCtZNPxev+Hf
+7tpT6sxDwDty4LHBUOZJG96N1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-
demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be A1206F460EXAMPLE
 REST.GET.BUCKETPOLICY - "GET /amzn-s3-demo-bucket1?policy HTTP/1.1" 404
 NoSuchBucketPolicy 297 - 38 - "-" "S3Console/0.4" - BNaBsXZQQDbssi6xMBdBU2sLt
+Yf5kZDmeBUP35sFoKa3sLLeMC78iwEIWxs99CRUrbS4n11234= SigV4 ECDHE-RSA-AES128-GCM-SHA256
 AuthHeader amzn-s3-demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - Yes
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:01:00 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 7B4A0FABBEXAMPLE
 REST.GET.VERSIONING - "GET /amzn-s3-demo-bucket1?versioning HTTP/1.1" 200 -
 113 - 33 - "-" "S3Console/0.4" - Ke1bUcazaN1jWuUlPJaxF64cQVpUEhoZKEG/hmy/gijN/
I1DeWqDfFvnpybfEseEME/u7ME1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-
demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - -
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:01:57 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DD6CC733AEXAMPLE REST.PUT.OBJECT s3-dg.pdf "PUT /amzn-s3-demo-bucket1/
s3-dg.pdf HTTP/1.1" 200 - - 4406583 41754 28 "-" "S3Console/0.4" -
 10S62Zv81kBW7BB6SX4XJ48o6kpcl6LPwEoizZQQxJd5qDSCTLX0TgS37kYUBKQW3+bPdrg1234= SigV4
 ECDHE-RSA-AES128-SHA AuthHeader amzn-s3-demo-bucket1.s3.us-west-1.amazonaws.com
 TLSV1.2 - Yes
```

> **Note**
>
> 任何字段都可以设置为 - 以指示数据未知或不可用，或者该字段不适用于此请求。

主题

- 日志记录字段

- 复制操作的其他日志记录
- 自定义访问日志信息
- 可扩展服务器访问日志格式的编程注意事项

日志记录字段

以下列表介绍了日志记录字段。

存储桶拥有者

源存储桶拥有者的规范用户 ID。规范用户 ID 是另一种形式的 AWS 账户 ID。有关规范用户 ID 的更多信息，请参阅《AWS 一般参考》中的 AWS 账户 标识符。有关如何查找您的账户的规范用户 ID 的信息，请参阅查找 AWS 账户 的规范用户 ID。

示例条目

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

存储桶

请求处理的存储桶的名称。如果系统收到格式错误的请求且无法确定存储桶，则请求不会显示在任何的服务器访问日志中。

示例条目

```

```

Time

收到请求的时间；这些日期和时间采用协调世界时 (UTC)。使用 `strftime()` 术语的格式如下所示：`[%d/%b/%Y:%H:%M:%S %z]`

示例条目

```
[06/Feb/2019:00:00:38 +0000]
```

远程 IP

请求者的显式 IP 地址。中间代理和防火墙可能会隐藏发出请求的计算机的实际 IP 地址。

示例条目

```
192.0.2.3
```

请求者

请求者的规范用户 ID 或用于未经验证请求的 -。如果请求者是 IAM 用户，此字段会返回请求者的 IAM 用户名以及该 IAM 用户所属的 AWS 账户根用户。此标识符与用于访问控制目的的标识符是相同的。

示例条目

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

如果请求者使用代入的角色，则此字段返回代入的 IAM 角色。

示例条目

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

请求 ID

由 Amazon S3 生成的字符串，可用于唯一地标识每个请求。

示例条目

```
3E57427F33A59F07
```

操作

此处列出的操作将声明为 SOAP.*operation*、REST.*HTTP_method*.*resource_type*、WEBSITE.*HTTP_method*.*resource_t* 或 BATCH.DELETE.OBJECT，或者 生命周期和日志记录 的 S3.action.resource_type。

示例条目

```
REST.PUT.OBJECT
```

键

请求的密钥（对象名称）部分。

示例条目

```
/photos/2019/08/puppy.jpg
```

Request-URI

HTTP 请求消息的 Request-URI 部分。

示例条目

```
"GET /amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-foo=bar HTTP/1.1"
```

HTTP 状态

响应的数字 HTTP 状态代码。

示例条目

```
200
```

错误代码

Amazon S3 错误代码；如果没有发生错误，则为 -。

示例条目

```
NoSuchBucket
```

发送的字节数

发送的响应字节数（不包括 HTTP 协议支出），或 -（如果为零)。

示例条目

```
2662992
```

对象大小

所涉及的对象的总大小。

示例条目

```
3462992
```

总时间

从服务器的角度传输请求的毫秒数。该值计算从收到请求到发送响应的最后一个字节的时间。由于网络延迟，从客户端计算出的时间可能会更长。

示例条目

```
70
```

周转时间

Amazon S3 处理您的请求所花费的毫秒数。该值计算从收到您的请求的最后一个字节到发出响应的第一个字节的时间。

示例条目

```
10
```

Referer

HTTP `Referer` 标头的值（如果存在）。发送请求时，HTTP 用户代理（例如，浏览器）通常会将此标头设置为链接的 URL 或嵌入页面。

示例条目

```
"http://www.example.com/webservices"
```

User-Agent

HTTP `User-Agent` 标头的值。

示例条目

```
"curl/7.15.1"
```

版本 ID

请求中的版本 ID；如果操作没有使用 versionId 参数，则为 -。

示例条目

```
3HL4kqtJvjVBH40Nrjfkd
```

主机 ID

x-amz-id-2 或 Amazon S3 扩展请求 ID。

示例条目

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

签名版本

签名版本，用于对请求进行身份验证的 SigV2 或 SigV4，或未经身份验证的请求的 -。

示例条目

```
SigV2
```

密码套件

协商用于发出 HTTPS 请求的安全套接字层（SSL）密码，对于 HTTP 为 -。

示例条目

```
ECDHE-RSA-AES128-GCM-SHA256
```

身份验证类型

所使用的请求身份验证的类型：身份验证标头使用 AuthHeader，查询字符串（预签名 URL）使用 QueryString，未经身份验证的请求使用 -。

示例条目

```
AuthHeader
```

主机标头

用于连接到 Amazon S3 的端点。

示例条目

```
s3.us-west-2.amazonaws.com
```

某些较早的区域支持传统的端点。您可能会在服务器访问日志或 AWS CloudTrail 日志中看到这些端点。有关更多信息，请参阅 传统终端节点。有关 Amazon S3 区域和端点的完整列表，请参阅《Amazon Web Services 一般参考》中的 Amazon S3 端点和限额。

TLS 版本

客户端协商的传输层安全性（TLS）版本。此值为以下值之一：TLSv1.1、TLSv1.2、TLSv1.3；如果不使用 TLS，则为 -。

示例条目

```
TLSv1.2
```

接入点 ARN

请求接入点的 Amazon Resource Name（ARN）。如果接入点的 ARN 格式不正确或未使用，则该字段将包含 -。有关接入点的更多信息，请参阅 使用接入点。有关 ARN 的更多信息，请参阅《AWS 参考指南》中的 Amazon 资源名称（ARN）。

示例条目

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

一个字符串，用于指示请求是否需要访问控制列表（ACL）以进行授权。如果请求需要 ACL 进行授权，则字符串为 Yes。如果不需要 ACL，则字符串为 -。有关 ACL 的更多信息，请参阅 访问控制列表 (ACL) 概述。有关使用 aclRequired 字段禁用 ACL 的更多信息，请参阅为您的存储桶控制对象所有权和禁用 ACL。。

示例条目

```
Yes
```

复制操作的其他日志记录

复制操作包括 GET 和 PUT。出于该原因，我们会在执行复制操作时记录两份记录。前面的部分描述了与操作的 PUT 部分相关的字段。以下列表描述了记录中与复制操作的 GET 部分相关的字段。

存储桶拥有者

用于存储将复制的对象的存储桶的规范用户 ID。规范用户 ID 是另一种形式的 AWS 账户 ID。有关规范用户 ID 的更多信息，请参阅《AWS 一般参考》中的 AWS 账户 标识符。有关如何查找您的账户的规范用户 ID 的信息，请参阅查找 AWS 账户 的规范用户 ID。

示例条目

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

存储桶

用于存储被复制对象的存储桶的名称。

示例条目

```

```

Time

收到请求的时间；这些日期和时间采用协调世界时 (UTC)。使用 `strftime()` 术语的格式如下所示：`[%d/%B/%Y:%H:%M:%S %z]`

示例条目

```
[06/Feb/2019:00:00:38 +0000]
```

远程 IP

请求者的显式 IP 地址。中间代理和防火墙可能会隐藏发出请求的计算机的实际 IP 地址。

示例条目

```
192.0.2.3
```

请求者

请求者的规范用户 ID 或用于未经验证请求的 -。如果请求者是 IAM 用户，此字段将返回请求者的 IAM 用户名以及该 IAM 用户所属的 AWS 账户根用户。此标识符与用于访问控制目的的标识符是相同的。

示例条目

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

如果请求者使用代入的角色，则此字段返回代入的 IAM 角色。

示例条目

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

请求 ID

由 Amazon S3 生成的字符串，可用于唯一地标识每个请求。

示例条目

```
3E57427F33A59F07
```

操作

此处列出的操作将声明为 SOAP.*operation*、REST.*HTTP_method*.*resource_type*、WEBSITE.*HTTP_method*.*resource_t* 或 BATCH.DELETE.OBJECT。

示例条目

```
REST.COPY.OBJECT_GET
```

键

被复制对象的键（对象名称）；或者，如果操作没有使用键参数，则为 -。

示例条目

```
/photos/2019/08/puppy.jpg
```

Request-URI

HTTP 请求消息的 Request-URI 部分。

示例条目

```
"GET /amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-foo=bar"
```

HTTP 状态

复制操作的 GET 部分的数字 HTTP 状态代码。

示例条目

```
200
```

错误代码

复制操作的 GET 部分的 Amazon S3 错误代码；或者，如果没有发生任何错误，则为 -。

示例条目

```
NoSuchBucket
```

发送的字节数

发送的响应字节数（不包括 HTTP 协议支出）；或者，如果为零，则为 -。

示例条目

```
2662992
```

对象大小

所涉及的对象的总大小。

示例条目

```
3462992
```

总时间

从服务器的角度传输请求的毫秒数。该值计算从收到请求到发送响应的最后一个字节的时间。由于网络延迟，从客户端计算出的时间可能会更长。

示例条目

```
70
```

周转时间

Amazon S3 处理您的请求所花费的毫秒数。该值计算从收到您的请求的最后一个字节到发出响应的第一个字节的时间。

示例条目

```
10
```

Referer

HTTP `Referer` 标头的值（如果存在）。发送请求时，HTTP 用户代理（例如，浏览器）通常会将此标头设置为链接的 URL 或嵌入页面。

示例条目

```
"http://www.example.com/webservices"
```

User-Agent

HTTP `User-Agent` 标头的值。

示例条目

```
"curl/7.15.1"
```

版本 ID

被复制对象的版本 ID，或者，如果 x-amz-copy-source 标头没有将 versionId 参数指定为复制源的一部分，则为 -。

示例条目

```
3HL4kqtJvjVBH40Nrjfkd
```

主机 ID

x-amz-id-2 或 Amazon S3 扩展请求 ID。

示例条目

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

签名版本

签名版本 SigV2 或 SigV4，用于对请求进行身份验证；或者，对于未经身份验证的请求，则为 -。

示例条目

```
SigV4
```

密码套件

协商用于发出 HTTPS 请求的安全套接字层（SSL）密码，或者，对于 HTTP 为 -。

示例条目

```
ECDHE-RSA-AES128-GCM-SHA256
```

身份验证类型

所使用的请求身份验证的类型：身份验证标头使用 AuthHeader，查询字符串（预签名 URL）使用 QueryString，未经身份验证的请求使用 -。

示例条目

```
AuthHeader
```

主机标头

用于连接到 Amazon S3 的端点。

示例条目

```
s3.us-west-2.amazonaws.com
```

某些较早的区域支持传统的端点。您可能会在服务器访问日志或 AWS CloudTrail 日志中看到这些端点。有关更多信息，请参阅 传统终端节点。有关 Amazon S3 区域和端点的完整列表，请参阅《Amazon Web Services 一般参考》中的 Amazon S3 端点和限额。

TLS 版本

客户端协商的传输层安全性（TLS）版本。此值为以下值之一：TLSv1.1、TLSv1.2、TLSv1.3；如果不使用 TLS，则为 -。

示例条目

```
TLSv1.2
```

接入点 ARN

请求接入点的 Amazon Resource Name（ARN）。如果接入点的 ARN 格式不正确或未使用，则该字段将包含 -。有关接入点的更多信息，请参阅 使用接入点。有关 ARN 的更多信息，请参阅《AWS 参考指南》中的 Amazon 资源名称（ARN）。

示例条目

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

一个字符串，用于指示请求是否需要访问控制列表（ACL）以进行授权。如果请求需要 ACL 进行授权，则字符串为 Yes。如果不需要 ACL，则字符串为 -。有关 ACL 的更多信息，请参阅 访问控制列表 (ACL) 概述。有关使用 aclRequired 字段禁用 ACL 的更多信息，请参阅为您的存储桶控制对象所有权和禁用 ACL。。

示例条目

```
Yes
```

自定义访问日志信息

您可以包含要存储在请求的访问日志记录中的自定义信息。为此，请将自定义查询字符串参数添加到请求的 URL 中。Amazon S3 忽略以 x- 开头的查询字符串参数，但是会将这些参数包含在请求的访问日志记录中，以作为日志记录的 Request-URI 字段的一部分。

例如，GET 的 "s3.amazonaws.com/amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-user=johndoe" 请求工作方式与 "s3.amazonaws.com/amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg" 的请求的相同，只是 "x-user=johndoe" 字符串包含在关联日志记录的 Request-URI 字段中。此功能仅在 REST 界面中可用。

可扩展服务器访问日志格式的编程注意事项

有时，我们可能会通过向每一行的末尾添加新字段来扩展访问日志记录格式。因此，请确保用于解析服务器访问日志的任何代码都可以处理它后续可能不理解的字段。

删除 Amazon S3 日志文件

一个已启用服务器访问日志记录的 Amazon S3 存储桶可能会随时间推移积累许多服务器日志对象。您的应用程序可能在创建之后的特定时间段内需要这些访问日志，在此之后，您可能希望删除它们。您可以使用 Amazon S3 生命周期配置来设置规则，以便 Amazon S3 可自动对这些对象进行排队，从而在其生命周期结束时进行删除。

您可以使用共享前缀为 S3 存储桶中的对象子集定义生命周期配置。如果您在服务器访问日志记录配置中指定了前缀，则可以设置生命周期配置规则，以删除具有该前缀的日志对象。

例如，假定您的日志对象具有前缀 logs/。您可以设置生命周期配置规则，以便在指定的时间段后删除存储桶中具有 logs/ 前缀的所有对象。

有关生命周期配置的更多信息，请参阅 管理存储生命周期。

有关服务器访问日志记录的一般信息，请参阅 使用服务器访问日志记录来记录请求。

使用 Amazon S3 服务器访问日志来确定请求

您可以使用 Amazon S3 服务器访问日志来确定 Amazon S3 请求。

> **Note**
>
> - 要确定 Amazon S3 请求，建议您使用 AWS CloudTrail 数据事件而不是 Amazon S3 服务器访问日志。CloudTrail 数据事件更易于设置并包含更多信息。有关更多信息，请参阅 使用 CloudTrail 识别 Amazon S3 请求。
> - 根据您获得的访问请求数量，与使用 CloudTrail 数据事件相比，分析日志使用的资源或时间可能会更多。

主题

使用 Amazon Athena 查询访问日志中的请求

您可以使用 Amazon Athena 查询 Amazon S3 访问日志以确定 Amazon S3 请求。

Amazon S3 将服务器访问日志作为对象存储在 S3 存储桶中。使用可以分析 Amazon S3 中的日志的工具通常会更轻松。Athena 支持分析 S3 对象，并且可用于查询 Amazon S3 访问日志。

Example

以下示例说明了如何在 Amazon Athena 中查询 Amazon S3 服务器访问日志。将以下示例中使用的 *user input placeholders* 替换为您自己的信息。

> **Note**
>
> 要在 Athena 查询中指定 Amazon S3 位置，您必须提供将日志传输到的存储桶的 S3 URI。此 URI 必须包含以下格式的存储桶名称和前缀：`s3://`*amzn-s3-demo-bucket1*`-logs/`*prefix*`/`

1. 从 https://console.aws.amazon.com/athena/ 打开 Athena 控制台。
2. 在查询编辑器中，运行类似如下的命令。将 *s3_access_logs_db* 替换为要为数据库指定的名称。

```
CREATE DATABASE s3_access_logs_db
```

> **Note**
>
> 最佳实践是在与 S3 存储桶所在相同的 AWS 区域中创建数据库。

3. 在查询编辑器中，运行类似如下的命令以便在步骤 2 中创建的数据库中创建一个表架构。将 *s3_access_logs_db.mybucket_logs* 替换为要为表指定的名称。STRING 和 BIGINT 数据类型值是访问日志属性。您可以在 Athena 中查询这些属性。对于 LOCATION，输入之前记下的 S3 存储桶和前缀。

```
CREATE EXTERNAL TABLE `s3_access_logs_db.mybucket_logs`(
  `bucketowner` STRING,
  `bucket_name` STRING,
  `requestdatetime` STRING,
  `remoteip` STRING,
  `requester` STRING,
  `requestid` STRING,
  `operation` STRING,
  `key` STRING,
  `request_uri` STRING,
  `httpstatus` STRING,
  `errorcode` STRING,
  `bytessent` BIGINT,
  `objectsize` BIGINT,
  `totaltime` STRING,
  `turnaroundtime` STRING,
  `referrer` STRING,
  `useragent` STRING,
  `versionid` STRING,
  `hostid` STRING,
  `sigv` STRING,
  `ciphersuite` STRING,
  `authtype` STRING,
  `endpoint` STRING,
  `tlsversion` STRING,
  `accesspointarn` STRING,
  `aclrequired` STRING)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
```

```
  'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) (-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 (\"[^\"]*\"|-) ([^ ]*)(?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*))?.*$')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://amzn-s3-demo-bucket1-logs/prefix/'
```

4. 在导航窗格中，在 Database (数据库) 下，请选择您的数据库。
5. 在 Tables (表) 下，请选择表名称旁边的 Preview table (预览表)。

   在 Results (结果) 窗格中，您应看到来自服务器访问日志中的数据，如 `bucketowner`、`bucket`、`requestdatetime` 等。这表示您成功创建了 Athena 表。您现在可以查询 Amazon S3 服务器访问日志。

Example — 显示已删除对象的人员和时间（时间戳、IP 地址和 IAM 用户）

```
SELECT requestdatetime, remoteip, requester, key
FROM s3_access_logs_db.mybucket_logs
WHERE key = 'images/picture.jpg' AND operation like '%DELETE%';
```

Example — 显示 IAM 用户执行的所有操作

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE requester='arn:aws:iam::123456789123:user/user_name';
```

Example — 显示在特定时间段内对对象执行的所有操作

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE Key='prefix/images/picture.jpg'
    AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
```

```
    BETWEEN parse_datetime('2017-02-18:07:00:00','yyyy-MM-dd:HH:mm:ss')
    AND parse_datetime('2017-02-18:08:00:00','yyyy-MM-dd:HH:mm:ss');
```

Example — 显示在特定时间段内传输到特定 IP 地址的数据量

```
SELECT coalesce(SUM(bytessent), 0) AS bytessenttotal
FROM s3_access_logs_db.mybucket_logs
WHERE remoteip='192.0.2.1'
AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-06-01','yyyy-MM-dd')
AND parse_datetime('2022-07-01','yyyy-MM-dd');
```

> **Note**
>
> 要减少保留日志的时间，您可以为服务器访问日志存储桶创建 S3 生命周期配置。创建生命周期配置规则以定期删除日志文件。这样做可以减少 Athena 为每个查询分析的数据量。有关更多信息，请参阅 在存储桶上设置生命周期配置。

使用 Amazon S3 访问日志确定签名版本 2 请求

对 Signature Version 2 的 Amazon S3 支持将会关闭（弃用）。之后，Amazon S3 将不再接受使用 Signature Version 2 的请求，并且所有请求必须使用 Signature Version 4 进行签署。您可以使用 Amazon S3 访问日志确定签名版本 2 访问请求。

> **Note**
>
> 要确定签名版本 2 请求，建议您使用 AWS CloudTrail 数据事件而不是 Amazon S3 服务器访问日志。CloudTrail 数据事件更易于设置，并且包含比服务器访问日志更多的信息。有关更多信息，请参阅 使用 CloudTrail 识别 Amazon S3 签名版本 2 请求。

Example — 显示发送签名版本 2 流量的所有请求者

```
SELECT requester, sigv, Count(sigv) as sigcount
FROM s3_access_logs_db.mybucket_logs
```

```
                    GROUP BY requester, sigv;
```

使用 Amazon S3 访问日志确定对象访问请求

对于诸如 GET、PUT 和 DELETE 等操作，您可以对 Amazon S3 服务访问日志使用查询以确定 Amazon S3 对象访问请求，并发现有关这些请求的进一步信息。

以下 Amazon Athena 查询示例说明如何从服务器访问日志中获取 Amazon S3 的所有 PUT 对象请求。

Example — 显示在特定时段内正在发送 **PUT** 对象请求的所有请求者

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db
WHERE operation='REST.PUT.OBJECT' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

以下 Amazon Athena 查询示例说明了如何从服务器访问日志中获取 Amazon S3 的所有 GET 对象请求。

Example — 显示在特定时段内正在发送 **GET** 对象请求的所有请求者

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
FROM s3_access_logs_db
WHERE operation='REST.GET.OBJECT' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

以下 Amazon Athena 查询示例说明了如何从服务器访问日志中获取向 S3 存储桶发出的所有匿名请求。

Example — 显示在特定时段内向存储桶发出请求的所有匿名请求者

```
SELECT bucket_name, requester, remoteip, key, httpstatus, errorcode, requestdatetime
```

```
FROM s3_access_logs_db.mybucket_logs
WHERE requester IS NULL AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2019-07-01:00:42:42','yyyy-MM-dd:HH:mm:ss')
AND
parse_datetime('2019-07-02:00:42:42','yyyy-MM-dd:HH:mm:ss')
```

以下 Amazon Athena 查询说明如何识别向 S3 存储桶发出的所有请求，这些请求需要访问控制列表（ACL）进行授权。您可以使用此信息将这些 ACL 权限迁移到相应的存储桶策略并禁用 ACL。创建这些存储桶策略后，您可以对这些存储桶禁用 ACL。有关禁用 ACL 的更多信息，请参阅禁用 ACL 的先决条件。

Example — 识别所有需要 ACL 进行授权的请求

```
SELECT bucket_name, requester, key, operation, aclrequired, requestdatetime
FROM s3_access_logs_db
WHERE aclrequired = 'Yes' AND
parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-05-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
AND parse_datetime('2022-08-10:00:00:00','yyyy-MM-dd:HH:mm:ss')
```

> **Note**
>
> - 您可以修改日期范围以满足您的需要。
> - 也可以使用这些查询示例进行安全监控。您可以查看意外或未经授权的 IP 地址或请求者发出的 `PutObject` 或 `GetObject` 调用的结果，以及确定向存储桶发出的任何匿名请求。
> - 此查询仅从启用了日志记录的时间检索信息。
> - 如果您使用 AWS CloudTrail 日志，请参阅 使用 CloudTrail 识别对 S3 对象的访问权限。

使用 Amazon CloudWatch 监控指标

Amazon S3 的 Amazon CloudWatch 指标可帮助您了解和提高使用 Amazon S3 的应用程序的性能。您可以通过多种方法将 CloudWatch 与 Amazon S3 结合使用。

存储桶的每日存储指标

使用 CloudWatch 来监控存储桶存储，此工具将从 Amazon S3 收集存储数据，并将数据处理为便于读取的每日指标。Amazon S3 的这些存储指标每天报告一次并提供给所有客户，无需额外费用。

请求指标

监控 Amazon S3 请求，以快速识别运行问题并对其采取措施。这些指标在要处理的某些延迟后每隔 1 分钟提供一次。这些 CloudWatch 指标的计费费率与 Amazon CloudWatch 自定义指标相同。有关 CloudWatch 定价的信息，请参阅 Amazon CloudWatch 定价。要了解如何选择获取这些指标，请参阅 CloudWatch 指标配置。

启用后，将为所有对象操作报告请求指标。默认情况下，这些 1 分钟指标在 Amazon S3 存储桶级别提供。您还可以使用共享前缀或对象标签或接入点为指标定义筛选条件。

- 接入点 - 接入点是命名为连接到存储桶的网络端点，可简化对 S3 中的共享数据集的大规模数据访问管理。通过接入点筛选条件，您可以深入了解您的接入点使用情况。有关接入点的更多信息，请参阅 监控和记录接入点。
- 前缀 – 尽管 Amazon S3 数据模型是一种扁平结构，但您仍可以使用前缀推断层次结构。前缀类似于允许您将存储桶中的类似对象组合在一起的目录名称。S3 控制台支持这些带有文件夹概念的前缀。如果您按前缀进行筛选，有相同前缀的对象将包含在指标配置中。有关前缀的更多信息，请参阅 使用前缀组织对象。
- 标签 – 标记是可以添加到对象的键值名称对。标签可让您轻松查找和整理对象。您还可以使用标签作为指标配置的筛选条件，以便仅包含具有这些标签的对象。有关对象标签的更多信息，请参阅 使用标签对存储进行分类。

要将这些指标与特定的业务应用程序、工作流或内部组织对齐，您可以在共享前缀、对象标记或接入点上进行筛选。

复制指标

复制指标 - 监控等待复制的 S3 API 操作的总数、等待复制的对象的总大小、到目标 AWS 区域的最长复制时间以及复制失败的操作总数。已启用 S3 Replication Time Control（S3 RTC）或 S3 复制指标的复制规则会发布复制指标。

有关更多信息，请参阅使用复制指标和 S3 事件通知监控进度或使用 S3 Replication Time Control（S3 RTC）满足合规性要求。

Amazon S3 Storage Lens 存储统计管理工具指标

您可以将 S3 Storage Lens 存储统计管理工具使用情况和活动指标发布到 Amazon CloudWatch，以便在 CloudWatch 控制面板中创建运营状况的统一视图。S3 Storage Lens 存储统计管理工具指

标可以在 AWS/S3/Storage-Lens 命名空间中可用。CloudWatch 发布选项可用于让 S3 Storage Lens 存储统计管理工具升级到 advanced metrics and recommendations（高级指标和建议）。您可以在 S3 Storage Lens 存储统计管理工具中为新的或现有的仪表板配置启用 CloudWatch 发布选项。

有关更多信息，请参阅 在 CloudWatch 中监控 S3 Storage Lens 存储统计管理工具指标。

所有 CloudWatch 统计数据会保留 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的运行情况。有关 CloudWatch 的更多信息，请参阅 Amazon CloudWatch 用户指南中的什么是 Amazon CloudWatch？。根据您的用例，您可能需要对 CloudWatch 警报进行一些额外的配置。例如，您可以使用指标数学表达式来创建警报。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的使用 CloudWatch 指标、使用指标数学、使用 Amazon CloudWatch 警报以及根据指标数学表达式创建 CloudWatch 警报。

最大努力 CloudWatch 指标传输

系统将以最大努力传输 CloudWatch 指标。大多数具有请求指标的针对 Amazon S3 对象的请求会导致将数据点发送到 CloudWatch。

无法保证指标的完整性和及时性。可能返回特定请求的数据点，其时间戳晚于实际处理请求的时间。1 分钟的数据点在通过 CloudWatch 提供之前可能会延迟，或者根本不会提供该数据点。您可以通过 CloudWatch 请求指标近乎实时地了解有关存储桶流量性质方面的信息。这并不意味着会完整记录所有请求。

根据此特征的最大努力性质，在账单和成本管理控制面板提供的报告中可能有一个或多个访问请求不会出现在存储桶指标中。

有关更多信息，请参阅以下主题。

主题

指标与维度

以下各表列出了 Amazon S3 发送到 Amazon CloudWatch 的存储指标和维度。

最大努力 CloudWatch 指标传输

系统将以最大努力传输 CloudWatch 指标。大多数具有请求指标的针对 Amazon S3 对象的请求会导致将数据点发送到 CloudWatch。

无法保证指标的完整性和及时性。可能返回特定请求的数据点，其时间戳晚于实际处理请求的时间。1 分钟的数据点在通过 CloudWatch 提供之前可能会延迟，或者根本不会提供该数据点。您可以通过 CloudWatch 请求指标近乎实时地了解有关存储桶流量性质方面的信息。这并不意味着会完整记录所有请求。

根据此特征的最大努力性质，在账单和成本管理控制面板提供的报告中可能有一个或多个访问请求不会出现在存储桶指标中。

主题

在 CloudWatch 中存储桶的 Amazon S3 每日存储指标

AWS/S3 命名空间包含存储桶的以下每日存储指标。

指标	描述
BucketSizeBytes	存储在以下存储类的存储桶中的数据量（以字节为单位）： • S3 Standard (STANDARD) • S3 Intelligent-Tiering (INTELLIGENT_TIERING) • S3 Standard-Infrequent Access (STANDARD_IA) • S3 One Zone-Infrequent Access (ONEZONE_IA)

指标	描述
	• 低冗余存储（RRS）（REDUCED_REDUNDANCY ） • S3 Glacier Instant Retrieval (GLACIER_IR) • S3 Glacier Deep Archive (DEEP_ARCHIVE) • S3 Glacier Flexible Retrieval (GLACIER) • S3 Express One Zone（EXPRESS_ONEZONE ） 此值通过汇总存储桶中所有对象（当前对象和非当前对象）和元数据（例如存储桶名称）的大小计算得出，包括所有向存储桶进行分段上传而未完成的所有部分的大小。 Note S3 Express One Zone 存储类仅可用于目录存储桶。 有效的存储类型筛选条件（请参阅 StorageType 维度） • S3 Standard：StandardStorage • S3 Intelligent-Tiering：IntelligentTieringFAStorage 、IntelligentTieringIAStorage 、IntelligentTieringAAStorage 、IntelligentTieringAIAStorage 、IntelligentTieringDAAStorage • S3 Standard-Infrequent Access：StandardIAStorage 、StandardIASizeOverhead 、StandardIAObjectOverhead • S3 One Zone-Infrequent Access：OneZoneIAStorage 、OneZoneIASizeOverhead • 低冗余存储（RRS）：ReducedRedundancyStorage • S3 Glacier Instant Retrieval：GlacierInstantRetrievalSizeOverhead 、GlacierInstantRetrievalStorage • S3 Glacier Flexible Retrieval：GlacierStorage 、GlacierStagingStorage 、GlacierObjectOverhead 、GlacierS3ObjectOverhead

指标	描述
	• S3 Glacier Deep Archive：DeepArchiveStorage 、DeepArchiveObjectOverhead 、DeepArchiveS3ObjectOverhead 、DeepArchiveStagingStorage • S3 Express One Zone：ExpressOneZoneStorage 单位：字节 有效统计数据：Average 有关 StorageType 维度的更多信息，请参阅 the section called “CloudWatch 中的 Amazon S3 维度”。
NumberOfObjects	通用存储桶中存储的所有存储类的对象的总数。此值通过对存储桶中的所有对象（包括当前对象和非当前对象）、删除标记以及所有向存储桶进行分段上传而未完成的所有分段的总数进行计数而计算得出。对于对象位于 S3 Express One Zone 存储类中的目录桶，此值通过对桶中所有对象进行计数来计算得出，但不包括向桶进行的多次未完成的上传次数。 有效的存储类型筛选条件：AllStorageTypes （请参阅 StorageType 维度） 单位：计数 有效统计数据：Average

CloudWatch 中的 Amazon S3 CloudWatch 请求指标

AWS/S3 命名空间包含以下请求指标。这些指标包括不可计费的请求（如果是来自 CopyObject 和复制的 GET 请求）。

> **Note**
>
> 目录桶不支持 CloudWatch 中的 Amazon S3 请求指标。

指标	描述
AllRequests	向 Amazon S3 存储桶提出的 HTTP 请求（不论类型如何）的总数。如果您要将某个指标配置用于某个筛选条件，则该指标将仅返回符合该筛选条件要求的 HTTP 请求。 单位：计数 有效统计数据：Sum
GetRequests	向 Amazon S3 存储桶中的对象发出的 HTTP GET 请求的数量。这不包括列表操作。对于每个 CopyObject 请求的来源，此指标会递增。 单位：计数 有效统计数据：Sum Note 面向分页列表的请求（例如 ListMultipartUploads、ListParts、ListObjectVersions 和其它）不包含在此指标中。
PutRequests	向 Amazon S3 存储桶中的对象发出的 HTTP PUT 请求的数量。对于每个 CopyObject 请求的目标，此指标会递增。 单位：计数 有效统计数据：Sum
DeleteRequests	向 Amazon S3 存储桶中的对象发出的 HTTP DELETE 请求的数量。此指标还包括 DeleteObjects 请求。此指标显示发出的请求数量，而不是删除的对象数量。 单位：计数 有效统计数据：Sum
HeadRequests	向 Amazon S3 存储桶发出的 HTTP HEAD 请求的数量。 单位：计数

指标	描述
	有效统计数据：Sum
PostRequests	向 Amazon S3 存储桶发出的 HTTP POST 请求的数量。 单位：计数 有效统计数据：Sum Note 此指标不包括 DeleteObjects 和 SelectObjectContent 请求。
SelectRequests	向 Amazon S3 存储桶中的对象发出的 Amazon S3 SelectObjectContent 请求的数量。 单位：计数 有效统计数据：Sum
SelectBytesScanned	使用 Amazon S3 存储桶中的 Amazon S3 SelectObjectContent 请求扫描的数据字节数。 单位：字节 有效统计数据：Average（每个请求的字节数）、Sum（每个周期的字节数）、Sample Count、Min、Max（与 p100 相同）、任何在 p0.0 和 p99.9 之间的百分位数
SelectBytesReturned	使用 Amazon S3 存储桶中的 Amazon S3 SelectObjectContent 请求返回的数据字节数。 单位：字节 有效统计数据：Average (每个请求的字节数)、Sum (每个周期的字节数)、Sample Count、Min、Max（与 p100 相同）、任何在 p0.0 和 p99.9 之间的百分位数

指标	描述
ListRequests	列出存储桶内容的 HTTP 请求的数量。 单位：计数 有效统计数据：Sum
BytesDownloaded	为向 Amazon S3 存储桶提出的请求下载的字节数（请求的响应包含正文）。 单位：字节 有效统计数据：Average（每个请求的字节数）、Sum（每个周期的字节数）、Sample Count、Min、Max（与 p100 相同）、任何在 p0.0 和 p99.9 之间的百分位数
BytesUploaded	为向 Amazon S3 存储桶发出的请求上传的字节数（请求包含正文）。 单位：字节 有效统计数据：Average (每个请求的字节数)、Sum (每个周期的字节数)、Sample Count、Min、Max（与 p100 相同）、任何在 p0.0 和 p99.9 之间的百分位数
4xxErrors	向 Amazon S3 存储桶发出的值为 0 或 1 的 HTTP 4xx 客户端错误状态代码请求数。Average 统计数据显示了错误率，Sum 统计数据显示了每个周期内该类型的错误的计数。 单位：计数 有效统计数据：Average (每个请求的报告数)、Sum (每个周期的报告数)、Min、Max、Sample Count

指标	描述
5xxErrors	向 Amazon S3 存储桶发出的值为 0 或 1 的 HTTP 5xx 服务器错误状态代码请求数。Average 统计数据显示了错误率，Sum 统计数据显示了每个周期内该类型的错误的计数。 单位：计数 有效统计数据：Average（每个请求的报告数）、Sum（每个周期的报告数）、Min、Max、Sample Count
FirstByteLatency	从 Amazon S3 存储桶收到完整请求到开始返回响应的每请求时间。 单位：毫秒 有效统计数据：Average、Sum、Min、Max（与 p100 相同）、Sample Count、任何在 p0.0 和 p100 之间的百分位数
TotalRequestLatency	从收到第一个字节到将最后一个字节发送到 Amazon S3 存储桶的已用每请求时间。此指标包括接收请求正文和发送响应正文所耗的时间（未包含在 FirstByteLatency 中）。 单位：毫秒 有效统计数据：Average、Sum、Min、Max（与 p100 相同）、Sample Count、任何在 p0.0 和 p100 之间的百分位数

CloudWatch 中的 S3 复制指标

您可以使用 S3 复制指标，通过跟踪待处理的字节、待处理的操作和复制延迟，监控复制的进度。有关更多信息，请参阅使用复制指标监控进度。

> **Note**
>
> 您可以在 Amazon CloudWatch 中对复制指标启用告警。为复制指标设置警报时，请将 Missing data treatment (丢失数据处理) 字段设置为 Treat missing data as ignore (maintain the alarm state) [将丢失的数据视为忽略（保持警报状态）]。

指标	描述
ReplicationLatency	对于给定的复制规则，复制目标 AWS 区域落后于源 AWS 区域的最大秒数。 单位：秒 有效统计数据：Max
BytesPendingReplication	给定复制规则的待复制对象的总字节数。 单位：字节 有效统计数据：Max
OperationsPendingReplication	给定复制规则的待复制操作数。 单位：计数 有效统计数据：Max
OperationsFailedReplication	给定复制规则的复制失败的操作数。 单位：计数 有效统计信息：总和（失败操作总数）、平均值（失败率）、样本数（复制操作总数）

CloudWatch 中的 S3 Storage Lens 存储统计管理工具指标

您可以将 S3 Storage Lens 存储统计管理工具使用情况和活动指标发布到 Amazon CloudWatch，以便在 CloudWatch 控制面板中创建运营状况的统一视图。S3 Storage Lens 存储统计管理工具指标已发布到 CloudWatch 中的 AWS/S3/Storage-Lens 命名空间。CloudWatch 发布选项可用于已升级为 advanced metrics and recommendations（高级指标和建议）的 S3 Storage Lens 存储统计管理工具控制面板。

有关发布到 CloudWatch 的 S3 Storage Lens 存储统计管理工具指标列表，请参阅 Amazon S3 Storage Lens 存储统计管理工具指标词汇表。有关维度的完整列表，请参阅 维度。

CloudWatch 中的 S3 对象 Lambda 请求指标

S3 对象 Lambda 包含以下请求指标。

指标	描述
`AllRequests`	使用对象 Lambda 接入点向 Amazon S3 存储桶发出的 HTTP 请求总数。 单位：计数 有效统计数据：Sum
`GetRequests`	使用对象 Lambda 接入点对于对象发出的 HTTP `GET` 请求的数量。该指标不包括列出操作。 单位：计数 有效统计数据：Sum
`BytesUploaded`	使用对象 Lambda 接入点上传到 Amazon S3 存储桶的字节数（请求包含正文）。 单位：字节 有效统计数据：Average（每个请求的字节数）、Sum（每个周期的字节数）、Sample Count、Min、Max（与 p100 相同）、任何在 p0.0 和 p99.9 之间的百分位数
`PostRequests`	使用对象 Lambda 接入点向 Amazon S3 存储桶发出的 HTTP `POST` 请求的数量。 单位：计数 有效统计数据：Sum
`PutRequests`	使用对象 Lambda 接入点对于 Amazon S3 存储桶中的对象发出的 HTTP PUT 请求的数量。 单位：计数 有效统计数据：Sum

指标	描述
DeleteRequests	使用对象 Lambda 接入点对于 Amazon S3 存储桶中的对象发出的 HTTP DELETE 请求的数量。此指标包括 DeleteObjects 请求。此指标显示发出的请求数量，而不是删除的对象数量。 单位：计数 有效统计数据：Sum
BytesDownloaded	为使用对象 Lambda 接入点向 Amazon S3 存储桶发出的请求下载的字节数（响应包括正文）。 单位：字节 有效统计数据：Average（每个请求的字节数）、Sum（每个周期的字节数）、Sample Count、Min、Max（与 p100 相同）、任何在 p0.0 和 p99.9 之间的百分位数
FirstByteLatency	从 Amazon S3 存储桶通过对象 Lambda 接入点收到完整请求到开始返回响应的每请求时间。此指标取决于在 AWS Lambda 函数将字节返回给对象 Lambda 接入点之前，该函数对于对象进行转换的运行时间。 单位：毫秒 有效统计数据：Average、Sum、Min、Max（与 p100 相同）、Sample Count、任何在 p0.0 和 p100 之间的百分位数
TotalRequestLatency	从收到第一个字节到将最后一个字节发送到对象 Lambda 接入点的已用每请求时间。此指标包括接收请求正文和发送响应正文所耗的时间（未包含在 FirstByteLatency 中）。 单位：毫秒 有效统计数据：Average、Sum、Min、Max（与 p100 相同）、Sample Count、任何在 p0.0 和 p100 之间的百分位数

指标	描述
HeadRequests	使用对象 Lambda 接入点向 Amazon S3 存储桶发出的 HTTP HEAD 请求的数量。 单位：计数 有效统计数据：Sum
ListRequests	列出 Amazon S3 存储桶内容的 HTTP GET 请求的数量。该指标同时包含 ListObjects 和 ListObjectsV2 操作。 单位：计数 有效统计数据：Sum
4xxErrors	使用对象 Lambda 接入点向 Amazon S3 存储桶发出的值为 0 或 1 的 HTTP 4xx 客户端错误状态代码请求数。Average 统计数据显示了错误率，Sum 统计数据显示了每个周期内该类型的错误的计数。 单位：计数 有效统计数据：Average（每个请求的报告数）、Sum（每个周期的报告数）、Min、Max、Sample Count
5xxErrors	使用对象 Lambda 接入点向 Amazon S3 存储桶发出的值为 0 或 1 的 HTTP 5xx 服务器错误状态代码请求数。Average 统计数据显示了错误率，Sum 统计数据显示了每个周期内该类型的错误的计数。 单位：计数 有效统计数据：Average（每个请求的报告数）、Sum（每个周期的报告数）、Min、Max、Sample Count
ProxiedRequests	向返回标准 Amazon S3 API 响应的对象 Lambda 接入点发出的 HTTP 请求的数量。（此类请求没有配置 Lambda 函数。） 单位：计数 有效统计数据：Sum

指标	描述
InvokedLambda	对在其中调用 Lambda 函数的 S3 对象的 HTTP 请求数。 单位：计数 有效统计数据：Sum
LambdaResponseRequests	Lambda 函数发出的 WriteGetObjectResponse 请求的数量。此指标仅适用于 GetObject 请求。
LambdaResponse4xx	从 Lambda 函数调用 WriteGetObjectResponse 时发生的 HTTP 4xx 客户端错误的数量。此指标提供的信息与 4xxErrors 相同，但仅适用于 WriteGetObjectResponse 调用。
LambdaResponse5xx	从 Lambda 函数调用 WriteGetObjectResponse 时发生的 HTTP 5xx 服务器错误的数量。此指标提供的信息与 5xxErrors 相同，但仅适用于 WriteGetObjectResponse 调用。

在 CloudWatch 中 Amazon S3 on Outposts 的指标

有关 CloudWatch 中用于 S3 on Outposts 的指标列表，请参阅CloudWatch 指标。

CloudWatch 中的 Amazon S3 维度

下列维度用于筛选 Amazon S3 指标。

维度	描述
BucketName	此维度筛选您仅为已识别存储桶请求的数据。
StorageType	此维度按以下存储类型筛选您存储在存储桶中的数据： • StandardStorage – 用于 STANDARD 存储类中的对象的字节数。 • IntelligentTieringAAStorage – 用于 INTELLIGENT_TIERING 存储类的归档访问层中的对象的字节数。 • IntelligentTieringAIAStorage – 用于 INTELLIGENT_TIERING 存储类的归档即时访问层中的对象的字节数。

维度	描述
	• IntelligentTieringDAAStorage – 用于 INTELLIGENT_TIERING 存储类的深层归档访问层中的对象的字节数。 • IntelligentTieringFAStorage – 用于 INTELLIGENT_TIERING 存储类的频繁访问层中的对象的字节数。 • IntelligentTieringIAStorage – 用于 INTELLIGENT_TIERING 存储类的不频繁访问层中的对象的字节数。 • StandardIAStorage – 用于 S3 Standard-Infrequent Access（STANDARD_IA）存储类中的对象的字节数。 • StandardIASizeOverhead – 用于 STANDARD_IA 存储类中小于 128KB 的对象的字节数。 • IntAAObjectOverhead – 对于归档访问层中 INTELLIGENT_TIERING 存储类的每个对象，S3 Glacier 为索引和相关元数据添加了 32KB 的存储空间。标识和还原对象需要此额外数据。按照 S3 Glacier Flexible Retrieval 费率对此附加存储收费。 • IntAAS3ObjectOverhead – 对于归档访问层中 INTELLIGENT_TIERING 存储类的每个对象，Amazon S3 使用 8KB 的存储空间作为对象和其他元数据的名称。将按照 S3 Standard 标准费率对此附加存储收费。 • IntDAAObjectOverhead – 对于深度归档访问层中 INTELLIGENT_TIERING 存储类的每个对象，S3 Glacier 为索引和相关元数据添加了 32KB 的存储空间。标识和还原对象需要此额外数据。按照 S3 Glacier Deep Archive 存储费率对此附加存储收费。 • IntDAAS3ObjectOverhead – 对于深度归档访问层中 INTELLIGENT_TIERING 存储类的每个对象，Amazon S3 为索引和相关元数据添加了 8KB 的存储空间。标识和还原对象需要此额外数据。将按照 S3 Standard 标准费率对此附加存储收费。 • OneZoneIAStorage – 用于“S3 单区 - 不频繁访问层（ONEZONE_IA）”存储类中的对象的字节数。 • OneZoneIASizeOverhead – 用于 ONEZONE_IA 存储类中小于 128KB 的对象的字节数。

维度	描述
	• ReducedRedundancyStorage – 用于低冗余存储 (RRS) 类中的对象的字节数。 • GlacierInstantRetrievalSizeOverhead – 用于 S3 Glacier Instant Retrieval 存储类中小于 128KB 的对象的字节数。 • GlacierInstantRetrievalStorage – 用于 S3 Glacier 即时检索存储类中的对象的字节数。 • GlacierStorage – 用于 S3 Glacier Flexible Retrieval 存储类中的对象的字节数。 • GlacierStagingStorage – 在 S3 Glacier Flexible Retrieval 存储类中的对象上完成 CompleteMultipartUpload 请求之前，用于多分段对象的各个分段的字节数。 • GlacierObjectOverhead – 对于每个归档对象，S3 Glacier 为索引及相关元数据添加 32KB 存储。标识和还原对象需要此额外数据。按照 S3 Glacier Flexible Retrieval 费率对此附加存储收费。 • GlacierS3ObjectOverhead – 对于归档到 S3 Glacier Flexible Retrieval 的每个对象，Amazon S3 将 8KB 存储用于对象和其他元数据的名称。将按照 S3 Standard 标准费率对此附加存储收费。 • DeepArchiveStorage – 用于 S3 Glacier Deep Archive 存储类中的对象的字节数。 • DeepArchiveObjectOverhead – 对于每个归档对象，S3 Glacier 为索引及相关元数据添加 32KB 存储。标识和还原对象需要此额外数据。按照 S3 Glacier Deep Archive 费率对此附加存储收费。 • DeepArchiveS3ObjectOverhead – 对于归档到 S3 Glacier Deep Archive 的每个对象，Amazon S3 将 8KB 存储用于对象和其他元数据的名称。将按照 S3 Standard 标准费率对此附加存储收费。 • DeepArchiveStagingStorage - 在 S3 Glacier Deep Archive 存储类中的对象上完成 CompleteMultipartUpload 请求之前，用于多分段对象的各个分段的字节数。

维度	描述
	• `ExpressOneZoneStorage` – 用于 S3 Express One Zone 存储类中的对象的字节数。
`FilterId`	此维度将筛选您为存储桶上的请求指标指定的指标配置。创建指标配置时，需要指定筛选条件 ID（例如，前缀、标签或接入点）。有关更多信息，请参阅创建指标配置。

CloudWatch 中的 S3 复制维度

下列维度用于筛选 S3 复制指标。

维度	描述
`SourceBucket`	从中进行复制的桶对象的名称。
`DestinationBucket`	要复制到的桶对象的名称。
`RuleId`	触发该复制指标进行更新的规则的唯一标识符。

CloudWatch 中的 S3 Storage Lens 存储统计管理工具维度

有关用于在 CloudWatch 中筛选 S3 Storage Lens 存储统计管理工具指标的维度列表，请参阅维度。

CloudWatch 中的 S3 对象 Lambda 请求维度

以下维度用于筛选来自对象 Lambda 接入点的数据。

维度	描述
`AccessPointName`	正在向其发出请求的接入点的名称。
`DataSourceARN`	对象 Lambda 接入点正从中检索数据的来源。如果请求调用 Lambda 函数，则这指的是 Lambda Amazon 资源名称（ARN）。否则，这指接入点 ARN。

访问 CloudWatch 指标

您可以按照以下步骤查看 Amazon S3 的存储指标。要获取涉及的 Amazon S3 指标，您必须设置开始和结束时间戳。对于任意给定 24 小时内的指标，请将时间段设置为 86400 秒（一天的秒数）。另外，请记得设置 BucketName 和 StorageType 维度。

使用 AWS CLI

例如，如果您想要使用 AWS CLI 以字节为单位获取特定桶的大小平均值，则可以使用以下命令：

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --namespace AWS/S3
 --start-time 2016-10-19T00:00:00Z --end-time 2016-10-20T00:00:00Z --statistics Average
 --unit Bytes --region us-west-2 --dimensions Name=BucketName,Value=amzn-s3-demo-bucket
 Name=StorageType,Value=StandardStorage --period 86400 --output json
```

此示例将生成以下输出。

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:00:00Z",
            "Average": 1025328.0,
            "Unit": "Bytes"
        }
    ],
    "Label": "BucketSizeBytes"
}
```

使用 S3 控制台

使用 Amazon CloudWatch 控制台查看指标

1. 访问 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在左侧导航窗格中，选择 Metrics。
3. 请选择 S3 命名空间。
4. （可选）要查看某个指标，请在搜索框中输入该指标的名称。
5. （可选）要按 StorageType 维度进行筛选，请在搜索框中输入该存储类的名称。

使用 AWS CLI 查看为您的 AWS 账户存储的有效指标列表

- 在命令提示符处，使用以下命令。

```
aws cloudwatch list-metrics --namespace "AWS/S3"
```

有关访问 CloudWatch 控制面板所需权限的更多信息，请参阅《Amazon CloudWatch 用户指南》中的 Amazon CloudWatch 控制面板权限。

CloudWatch 指标配置

借助 Amazon S3 的 Amazon CloudWatch 请求指标，您可以接收 1 分钟 CloudWatch 指标、设置 CloudWatch 警报和访问 CloudWatch 控制面板，从而近乎实时地查看 Amazon S3 存储的运行和性能。对于依赖于云存储的应用程序而言，这些指标使您可以快速识别运行问题并采取措施。启用后，这些 1 分钟指标默认在 Amazon S3 存储桶级别提供。

如果您要获取某个存储桶中的对象的 CloudWatch 请求指标，则必须为该存储桶创建指标配置。有关更多信息，请参阅 为存储桶中的所有对象创建 CloudWatch 指标配置。

您还可以使用共享前缀，对象标签或接入点为收集的指标定义筛选条件。这种定义筛选条件的方法将使您能够使各个指标筛选条件满足特定业务应用程序、工作流或内部组织的要求。有关更多信息，请参阅创建按前缀，对象标签或接入点筛选的指标配置。有关可用 CloudWatch 指标的更多信息和存储指标与请求指标之间的区别，请参阅 使用 Amazon CloudWatch 监控指标。

在使用指标配置时，请记住以下几点：

- 每个存储桶最多可以配置 1,000 个指标。
- 您可以利用筛选条件选择存储桶中的哪些对象可纳入指标配置。您可以筛选共享前缀，对象标签或接入点使各个指标筛选条件满足特定业务应用程序、工作流或内部组织的需求。要请求整个存储桶的指标，请创建一种不带筛选条件的指标配置。
- 只有启用请求指标才需要指标配置。存储桶级每日存储指标始终保持开启状态，并且免费提供。目前不能为筛选过的对象子集获取日常存储指标。
- 每个指标配置均可启用全套可用请求指标。只有存在适用于您的存储桶或筛选条件请求类型，才会报告特定于操作的指标（例如 `PostRequests`）。
- 将为对象级别操作报告请求指标。还为列出存储桶内容的操作 GET Bucket（List Objects）、GET Bucket Object Versions 和 List Multipart Uploads 报告请求指标，但不会为存储桶上的其他操作进行报告。

- 请求指标支持按前缀，对象标签或接入点进行筛选，但存储指标不支持。

最大努力 CloudWatch 指标传输

系统将以最大努力传输 CloudWatch 指标。大多数具有请求指标的针对 Amazon S3 对象的请求会导致将数据点发送到 CloudWatch。

无法保证指标的完整性和及时性。可能返回特定请求的数据点，其时间戳晚于实际处理请求的时间。1 分钟的数据点在通过 CloudWatch 提供之前可能会延迟，或者根本不会提供该数据点。您可以通过 CloudWatch 请求指标近乎实时地了解有关存储桶流量性质方面的信息。这并不意味着会完整记录所有请求。

根据此特征的最大努力性质，在账单和成本管理控制面板提供的报告中可能有一个或多个访问请求不会出现在存储桶指标中。

有关在 Amazon S3 中使用 CloudWatch 指标的更多信息，请参阅以下主题。

主题

- 为存储桶中的所有对象创建 CloudWatch 指标配置
- 创建按前缀，对象标签或接入点筛选的指标配置
- 删除指标筛选条件

为存储桶中的所有对象创建 CloudWatch 指标配置

配置请求指标时，您可以为存储桶中的所有对象创建 CloudWatch 指标配置，也可以按前缀，对象标签或接入点进行筛选。本主题中的步骤向您介绍如何为存储桶中的所有对象创建配置。要创建按对象标签，前缀或接入点筛选的配置，请参阅 创建按前缀，对象标签或接入点筛选的指标配置。

Amazon S3 存在三种类型的 Amazon CloudWatch 指标：存储指标、请求指标和复制指标。存储指标每天报告一次并提供给所有客户，无需额外费用。请求指标在要处理的某些延迟后每隔一分钟提供一次。请求指标按标准 CloudWatch 费率计费。您必须通过在控制台中配置请求指标或使用 Amazon S3 API 来选择使用请求指标。S3 复制指标为复制配置中的复制规则提供了详细的指标。使用复制指标，您可以通过跟踪待复制的字节数、待复制的操作数、复制失败的操作数和复制延迟来逐分钟监控进度。

有关 Amazon S3 的 CloudWatch 指标的更多信息，请参阅 使用 Amazon CloudWatch 监控指标。

您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）或 Amazon S3 REST API 向存储桶添加指标配置。

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择您要为其获取请求指标的对象所在的存储桶的名称。
3. 请选择 Metrics（指标）选项卡。
4. 在 Bucket metrics（存储桶指标）下，请选择 View additional charts（查看其他图表）。
5. 请选择 Request metrics（请求指标）选项卡。
6. 请选择 Create filter（创建筛选器）。
7. 在 Filter name（筛选器名称）框中，输入筛选器名称。

   名称可以包含字母、数字、句点、短划线和下划线。建议对应用于所有对象的筛选器使用 `EntireBucket` 作为名称。
8. 在选择筛选范围下，请选择此筛选条件应用于存储桶中的所有对象。

   您还可以定义筛选器，以便仅对此存储桶中的一部分对象收集和报告指标。有关更多信息，请参阅创建按前缀，对象标签或接入点筛选的指标配置。
9. 选择 Save changes (保存更改)。
10. 在 Request metrics (请求指标) 选项卡的 Filters (筛选器) 下，请选择刚创建的筛选器。

    大约 15 分钟后，CloudWatch 开始跟踪这些请求指标。您可以在 Request metrics (请求指标) 选项卡上查看它们。您可以在 Amazon S3 或 CloudWatch 控制台上查看指标的图形。请求指标按标准 CloudWatch 费率计费。有关更多信息，请参阅 Amazon CloudWatch 定价。

使用 REST API

您还可以使用 Amazon S3 REST API 以编程方式添加指标配置。有关添加和使用指标配置的更多信息，请参阅 Amazon Simple Storage Service API 参考中的以下主题：

- PUT Bucket Metric 配置
- GET Bucket Metric 配置
- List Bucket Metric 配置
- DELETE Bucket Metric 配置

使用 AWS CLI

1. 安装并设置 AWS CLI。有关说明，请参阅 AWS Command Line Interface 用户指南中的安装、更新和卸载 AWS CLI。
2. 打开终端。
3. 运行以下命令以添加指标配置。

```
aws s3api put-bucket-metrics-configuration --endpoint https://s3.us-
west-2.amazonaws.com --bucket bucket-name --id metrics-config-id --metrics-
configuration '{"Id":"metrics-config-id"}'
```

创建按前缀，对象标签或接入点筛选的指标配置

Amazon S3 存在三种类型的 Amazon CloudWatch 指标：存储指标、请求指标和复制指标。存储指标每天报告一次并提供给所有客户，无需额外费用。请求指标在要处理的某些延迟后每隔一分钟提供一次。请求指标按标准 CloudWatch 费率计费。您必须通过在控制台中配置请求指标或使用 Amazon S3 API 来选择使用请求指标。S3 复制指标为复制配置中的复制规则提供了详细的指标。使用复制指标，您可以通过跟踪待复制的字节数、待复制的操作数、复制失败的操作数和复制延迟来逐分钟监控进度。

有关 Amazon S3 的 CloudWatch 指标的更多信息，请参阅 使用 Amazon CloudWatch 监控指标。

配置 CloudWatch 指标时，您可以为存储桶中的所有对象创建筛选条件，也可以将配置筛选到单个存储桶中的相关对象组中。您可以基于以下一个或多个筛选条件类型在存储桶中筛选包含在指标配置中的对象：

- 对象键名称前缀 – 尽管 Amazon S3 数据模型是一种扁平结构，但您仍可以使用前缀推断层次结构。Amazon S3 控制台支持这些带有文件夹概念的前缀。如果您按前缀进行筛选，有相同前缀的对象将包含在指标配置中。有关前缀的更多信息，请参阅 使用前缀组织对象。
- 标签 – 您可以为对象添加标签（键值名称对）。标签可让您轻松查找和整理对象。您还可以使用标签作为指标配置的筛选条件。有关对象标签的更多信息，请参阅 使用标签对存储进行分类。
- 接入点 - S3 接入点是命名为连接到存储桶的网络端点，可简化对 S3 中的共享数据集的大规模数据访问管理。创建接入点筛选条件时，Amazon S3 会包含对指标配置中指定的接入点请求。有关更多信息，请参阅 监控和记录接入点。

> **Note**
>
> 创建按接入点筛选的指标配置时，必须使用接入点 Amazon Resource Name (ARN)，而不是接入点别名。请确保您将 ARN 用于接入点本身，而不是特定对象的 ARN。有关接入点 ARN 的更多信息，请参阅 使用接入点。

如果您指定了一项筛选条件，则仅在单个对象运行的请求可以满足筛选条件，并包含在报告的指标中。与 DeleteObjects 和 `ListObjects` 请求类似的请求不会使用筛选条件为配置返回任何指标。

要请求更复杂的筛选，请选择两个或更多元素。只有拥有所有这些元素的对象才会包含在指标配置中。如果未设置筛选条件，则存储桶中的所有对象都会包含在指标配置中。

使用 S3 控制台

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择您要为其获取请求指标的对象所在的存储桶的名称。
3. 请选择 Metrics (指标) 选项卡。
4. 在 Bucket metrics（存储桶指标）下，请选择 View additional charts（查看其他图表）。
5. 请选择 Request metrics（请求指标）选项卡。
6. 请选择 Create filter（创建筛选器）。
7. 在 Filter name（筛选器名称）框中，输入筛选器名称。

   名称可以包含字母、数字、句点、短划线和下划线。
8. 在筛选条件范围中，请选择使用前缀、对象标签和 S3 接入点（或三个全选）来限制此筛选条件的范围。
9. 在筛选条件类型中，请至少选择一个筛选条件类型：前缀、对象标签，或接入点。
10. 要定义前缀筛选条件并将筛选范围限制为单个路径，请在前缀框中输入前缀。
11. 要定义对象标签筛选条件，请在对象标签中，请选择添加标签，然后输入标签密钥和值。
12. 要定义接入点筛选条件，请在 S3 接入点字段中，输入接入点 ARN，或选择浏览 S3导航到接入点。

> **Important**
>
> 您无法输入接入点别名。必须输入接入点本身的 ARN，而不是特定对象的 ARN。

13. 选择 Save changes (保存更改)。

    Amazon S3 会用您指定的前缀，标签或接入点创建筛选条件。

14. 在 Request metrics（请求指标）选项卡的 Filters（筛选器）下，请选择刚创建的筛选器。

    您现在已创建一个筛选条件，该筛选条件通过前缀，标签或接入点限制请求指标范围。在 CloudWatch 开始跟踪这些请求指标后大约 15 分钟，您就可以看到 Amazon S3 和 CloudWatch 控制台中的指标的图表。请求指标按标准 CloudWatch 费率计费。有关更多信息，请参阅 Amazon CloudWatch 定价。

    还可以在存储桶级别配置请求指标。有关信息，请参阅为存储桶中的所有对象创建 CloudWatch 指标配置。

使用 AWS CLI

1. 安装并设置 AWS CLI。有关说明，请参阅 AWS Command Line Interface 用户指南中的安装、更新和卸载 AWS CLI。
2. 打开终端。
3. 要添加指标配置，运行以下命令。

   Example ：按前缀筛选

   ```
   aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
   id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
   {"Prefix":"prefix1"}} '
   ```

   Example ：按标签筛选

   ```
   aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
   id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
   {"Tag": {"Key": "string", "Value": "string"}} '
   ```

Example ：按接入点筛选

```
aws s3api put-bucket-metrics-configuration --bucket DOC-EXAMPLE-BUCKET1 --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"AccessPointArn":"arn:aws:s3:Region:account-id:accesspoint/access-point-name"}} '
```

Example ：按前缀、标签和接入点筛选

```
aws s3api put-bucket-metrics-configuration --endpoint https://
s3.Region.amazonaws.com --bucket DOC-EXAMPLE-BUCKET1 --id metrics-config-id --
metrics-configuration '
{
    "Id": "metrics-config-id",
    "Filter": {
        "And": {
            "Prefix": "string",
            "Tags": [
                {
                    "Key": "string",
                    "Value": "string"
                }
            ],
            "AccessPointArn": "arn:aws:s3:Region:account-id:accesspoint/access-
point-name"
        }
    }
}'
```

使用 REST API

您还可以使用 Amazon S3 REST API 以编程方式添加指标配置。有关添加和使用指标配置的更多信息，请参阅 Amazon Simple Storage Service API 参考中的以下主题：

- PUT Bucket Metric 配置
- GET Bucket Metric 配置
- List Bucket Metric 配置
- DELETE Bucket Metric 配置

删除指标筛选条件

如果您不再需要 Amazon CloudWatch 请求指标筛选器，可以删除它。删除筛选条件时，您不再需要为使用该特定筛选条件的请求指标付费。但是，您将继续为存在的任何其他筛选条件配置付费。

删除筛选条件时，您不再能够对请求指标使用筛选条件。删除筛选条件的操作无法撤消。

有关创建请求指标筛选条件的信息，请参阅以下主题：

- 为存储桶中的所有对象创建 CloudWatch 指标配置
- 创建按前缀，对象标签或接入点筛选的指标配置

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket（存储桶）列表中，请选择您的存储桶名称。
3. 请选择 Metrics（指标）选项卡。
4. 在 Bucket metrics（存储桶指标）下，请选择 View additional charts（查看其他图表）。
5. 请选择 Request metrics (请求指标) 选项卡。
6. 请选择 Manage filters (管理筛选条件)。
7. 请选择您的筛选条件。

   > **Important**
   >
   > 删除筛选条件的操作无法撤消。

8. 请选择 Delete。

   Amazon S3 会删除您的筛选条件。

使用 REST API

您还可以使用 Amazon S3 REST API 以编程方式添加指标配置。有关添加和使用指标配置的更多信息，请参阅 Amazon Simple Storage Service API 参考中的以下主题：

- PUT Bucket Metric 配置

- GET Bucket Metric 配置
- List Bucket Metric 配置
- DELETE Bucket Metric 配置

Amazon S3 事件通知

您可以使用 Amazon S3 事件通知特征在 S3 存储桶中发生某些事件时接收通知。要启用通知，请添加一个通知配置，该配置标识您希望 Amazon S3 发布的事件。确保它还标识您希望 Amazon S3 将通知发送到的目的地。您可以将此配置存储在与存储桶关联的通知子资源中。有关更多信息，请参阅 存储桶配置选项。Amazon S3 提供了一个 API，用于管理此子资源。

> **Important**
>
> Amazon S3 事件通知设计为至少传送一次。通常，事件通知在几秒内传输，不过有时可能需要一分钟或更长时间。

Amazon S3 事件通知概述

目前，Amazon S3 可以发布用于以下事件的通知：

- 新的对象创建事件
- 对象移除事件
- 还原对象事件
- 低冗余存储 (RRS) 对象丢失事件
- 复制事件
- S3 生命周期到期事件
- S3 生命周期转换事件
- S3 Intelligent-Tiering 自动归档事件
- 对象标记事件
- 对象 ACL PUT 事件

有关受支持事件类型的完整描述，请参阅 SQS、SNS 和 Lambda 支持的事件类型。

Amazon S3 可以将事件通知消息发送到以下目标。您在通知配置中指定这些目标的 Amazon Resource Name (ARN) 值。

- Amazon Simple Notification Service (Amazon SNS) 主题
- Amazon Simple Queue Service (Amazon SQS) 队列
- AWS Lambda 函数
- Amazon EventBridge

有关更多信息，请参阅 受支持的事件目标。

> **Note**
>
> 不支持将 Amazon Simple Queue Service FIFO（先进先出）队列作为 Amazon S3 事件通知目标。要向 Amazon SQS FIFO 队列发送 Amazon S3 事件的通知，您可以使用 Amazon EventBridge。有关更多信息，请参阅 启用 Amazon EventBridge。

> **Warning**
>
> 如果您的通知写入触发通知的同一存储桶，则可能会导致执行循环。例如，如果每当上传一个对象，存储桶就触发某个 Lambda 函数，而该函数又上传一个对象给存储桶，则该函数间接触发了自身。为避免这种情况，请使用两个存储桶，或将触发器配置为仅适用于传入对象所用的前缀。
>
> 有关将 Amazon S3 通知与 AWS Lambda 结合使用的更多信息和示例，请参阅 AWS Lambda 开发人员指南中的结合使用 AWS Lambda 和 Amazon S3。

有关每个存储桶可以创建的事件通知配置数的更多信息，请参阅 AWS 一般参考中的 Amazon S3 服务配额。

有关事件通知的更多信息，请参阅以下部分：

主题

- 事件通知类型和目标
- 使用 Amazon SQS、Amazon SNS 和 Lambda
- 使用 EventBridge

事件通知类型和目标

Amazon S3 支持多种可以发布通知的事件通知类型和目标。配置事件通知时，您可以指定事件类型和目标。只能为每个事件通知指定一个目标。Amazon S3 事件通知为每条通知消息发送一个事件条目。

主题

受支持的事件目标

Amazon S3 可以将事件通知消息发送到以下目标。

- Amazon Simple Notification Service（Amazon SNS）主题
- Amazon Simple Queue Service（Amazon SQS）队列
- AWS Lambda
- Amazon EventBridge

但是，只能为每个事件通知指定一种目标类型。

> **Note**
>
> 您必须授予 Amazon S3 将消息发布到 Amazon SNS 主题或 Amazon SQS 队列的权限。您还必须授予 Amazon S3 代表您调用 AWS Lambda 函数的权限。有关如何授予这些权限的说明，请参阅 授予将事件通知消息发布到目标的权限。

Amazon SNS 主题

Amazon SNS 是一项灵活且完全托管的消息推送服务。使用此服务，您可以将消息推送到移动设备或分布式服务。通过 SNS，，您只需要发布一次消息，就能将其发送无数遍。目前，仅允许标准 SNS 作为 S3 事件通知目标，而不允许 SNS FIFO。

Amazon SNS 用于协调和管理向订阅端点或客户交付或发送消息的过程。您可以使用 Amazon SNS 控制台创建 Amazon SNS 主题以便向其发送通知。

该主题必须与您的 Amazon S3 存储桶位于同一 AWS 区域。有关如何创建 Amazon SNS 主题的信息，请参阅 Amazon Simple Notification Service 开发人员指南中的 Amazon SNS 入门和 Amazon SNS 常见问题。

您先需要以下内容，然后才能将创建的 Amazon SNS 主题用作事件通知目标：

- Amazon SNS 主题的 Amazon Resource Name (ARN)
- 有效的 Amazon SNS 主题订阅。当消息发布到您的 Amazon SNS 主题时，主题订阅者将得到通知。

Amazon SQS 队列

Amazon SQS 提供了可靠且可扩展的托管队列，用于存储计算机之间传输的消息。您可以使用 Amazon SQS 来传输任何容量的数据，而不需要其他服务始终可用。您可以使用 Amazon SQS 控制台创建 Amazon SQS 队列以便向其发送通知。

Amazon SQS 队列必须与您的 Amazon S3 存储桶位于同一 AWS 区域。有关如何创建 Amazon SQS 队列的说明，请参阅 Amazon Simple Queue Service 开发人员指南中的什么是 Amazon Simple Queue Service 和 Amazon SQS 入门。

您先需要以下内容，然后才能使用 Amazon SQS 队列作为事件通知目标：

- Amazon SQS 队列的 Amazon Resource Name (ARN)

> **Note**
>
> 不支持将 Amazon Simple Queue Service FIFO（先进先出）队列作为 Amazon S3 事件通知目标。要向 Amazon SQS FIFO 队列发送 Amazon S3 事件的通知，您可以使用 Amazon EventBridge。有关更多信息，请参阅 启用 Amazon EventBridge。

Lambda 函数

您可以使用 AWS Lambda 通过自定义逻辑扩展其他 AWS 服务，或创建您自己的按 AWS 规模、性能和安全性运行的后端。借助 Lambda，您可以创建仅在需要时运行的离散、事件驱动的应用程序。您还可以使用它自动将这些应用程序从每天的几个请求扩展到每秒数千个请求。

Lambda 可以运行自定义代码以响应 Amazon S3 存储桶事件。您将自定义代码上传到 Lambda，并创建所谓的 Lambda 函数。当 Amazon S3 检测到特定类型的事件时，它可以将该事件发布到 AWS

Lambda 并在 Lambda 中调用您的函数。作为回应，Lambda 将运行您的函数。例如，它可能检测到的一种事件类型是对象创建的事件。

您可以使用 AWS Lambda 控制台创建使用 AWS 基础设施代表您运行代码的 Lambda 函数。Lambda 函数必须与您的 S3 存储桶位于同一区域。您还必须具有 Lambda 函数的名称或 ARN 才能将 Lambda 函数设置为事件通知目标。

> **Warning**
>
> 如果您的通知写入触发通知的同一存储桶，则可能会导致执行循环。例如，如果每当上传一个对象，存储桶就触发某个 Lambda 函数，而该函数又上传一个对象给存储桶，则该函数间接触发了自身。为避免这种情况，请使用两个存储桶，或将触发器配置为仅适用于传入对象所用的前缀。
>
> 有关将 Amazon S3 通知与 AWS Lambda 结合使用的更多信息和示例，请参阅 AWS Lambda 开发人员指南中的结合使用 AWS Lambda 和 Amazon S3。

Amazon EventBridge

Amazon EventBridge 是一个无服务器事件总线，它接收来自 AWS 服务的事件。您可以设置规则来匹配事件并将它们传输到目标，例如 AWS 服务或 HTTP 端点。有关更多信息，请参阅 Amazon EventBridge 用户指南中的什么是 EventBridge。

与其他目的地不同，您可以启用或禁用为存储桶传送到 EventBridge 的事件。如果启用传输，所有事件都发送到 EventBridge。此外，您可以使用 EventBridge 规则将事件路由到其他目标。

SQS、SNS 和 Lambda 支持的事件类型

Amazon S3 可以发布以下类型的事件。您在通知配置中指定这些事件类型。

事件类型	描述
s3:TestEvent	启用通知后，Amazon S3 将发布测试通知。这是为了确保主题存在，并且存储桶拥有者有权发布指定主题。 如果启用通知失败，则不会收到测试通知。
s3:ObjectCreated:* s3:ObjectCreated:Put	诸如 PUT、POST 和 COPY 之类的 Amazon S3 API 操作可以创建对象。通过这些事件类型，您可以在使用特定 API 操作创建对象时启用通知。或者，您可以使用 `s3:Object`

事件类型	描述
s3:ObjectCreated:Post s3:ObjectCreated:Copy s3:ObjectCreated:CompleteMultipartUpload	`Created:*` 事件类型来请求通知，而不考虑用于创建对象的 API。 `s3:ObjectCreated:CompleteMultipartUpload` 包含使用 UploadPartCopy复制操作创建的对象。
s3:ObjectRemoved:* s3:ObjectRemoved:Delete s3:ObjectRemoved:DeleteMarkerCreated	通过使用 ObjectRemoved 事件类型，您可以在从存储桶中删除一个对象或一批对象时启用通知。 您可以使用 `s3:ObjectRemoved:Delete` 事件类型请求在删除对象或永久删除受版本控制的对象时收到通知。或者，也可以使用 `s3:ObjectRemoved:DeleteMarkerCreated` 请求在为受版本控制的对象创建删除标记时收到通知。有关如何删除版本控制对象的说明，请参阅 从启用了版本控制的存储桶中删除对象版本。您还可以使用通配符 `s3:ObjectRemoved:*` 来请求在每次删除对象时收到通知。 这些事件通知不会针对从生命周期配置中的自动删除或失败的操作向您发出提示。
s3:ObjectRestore:* s3:ObjectRestore:Post s3:ObjectRestore:Completed s3:ObjectRestore:Delete	通过使用 ObjectRestore 事件类型，可在从 S3 Glacier Flexible Retrieval 存储类、S3 Glacier Deep Archive 存储类、S3 Intelligent-Tiering 存档访问层和 S3 Intelligent-Tiering 深度存档访问层还原对象时收到事件启动和完成的通知。您还可以接收对象的恢复副本何时过期的通知。 `s3:ObjectRestore:Post` 事件类型通知您对象还原启动。`s3:ObjectRestore:Completed` 事件类型会通知您还原完成。`s3:ObjectRestore:Delete` 事件类型会在恢复的对象的临时副本过期时通知您。
s3:ReducedRedundancyLostObject	当 Amazon S3 检测到 RRS 存储类的对象丢失时，您会收到此通知事件。

事件类型	描述
s3:Replication:* s3:Replication:OperationFailedReplication s3:Replication:OperationMissedThreshold s3:Replication:OperationReplicatedAfterThreshold s3:Replication:OperationNotTracked	通过使用 Replication（复制）事件类型，您可以对具有 S3 复制指标或启用 S3 Replication Time Control（S3 RTC）的复制配置接收通知。您可以通过跟踪待处理字节数、待处理的操作和复制延迟来逐分钟监控复制事件的进度。有关复制指标的信息，请参阅 使用复制指标和 S3 事件通知监控进度。 当符合复制条件的对象复制失败时，`s3:Replication:OperationFailedReplication` 事件类型将通知您。当符合复制条件的对象超过 15 分钟的复制阈值时，`s3:Replication:OperationMissedThreshold` 事件类型将通知您。 当一个使用 S3 复制时间控件的复制符合条件的对象在 15 分钟阈值后复制时，`s3:Replication:OperationReplicatedAfterThreshold` 事件类型就会通知您。`s3:Replication:OperationNotTracked` 事件类型在使用 S3 复制时间控件但不再被复制指标跟踪的符合复制资格的对象时通知您。
s3:LifecycleExpiration:* s3:LifecycleExpiration:Delete s3:LifecycleExpiration:DeleteMarkerCreated	通过使用 LifecycleExpiration 事件类型，当 Amazon S3 根据 S3 生命周期配置删除对象时，您可以收到通知。 `s3:LifecycleExpiration:Delete` 事件类型在删除未转换存储桶中的对象时通知您。当 S3 生命周期配置永久删除对象版本时，它也会通知您。版本控制的存储桶中对象的当前版本删除后，当 S3 生命周期创建删除标记时，`s3:LifecycleExpiration:DeleteMarkerCreated` 事件类型会通知您。
s3:LifecycleTransition	当一个对象通过 S3 Lifecycle 配置转移到另一个 Amazon S3 存储类时，您会收到这个通知事件。
s3:IntelligentTiering	当 S3 Intelligent-Tiering 存储类中的对象迁移到归档访问层或深层归档访问层时，会收到此通知事件。

事件类型	描述
s3:ObjectTagging:* s3:ObjectTagging:Put s3:ObjectTagging:Delete	通过使用 ObjectTagging 事件类型，您可以在对象标签从对象中添加或删除时启用通知。 `s3:ObjectTagging:Put` 事件类型会在对象上放置标签或现有标签更新时通知您。当从对象中移除标签时，`s3:ObjectTagging:Delete` 事件类型会通知您。
s3:ObjectAcl:Put	当 ACL 放在对象上或更改现有 ACL 时，您会收到此通知事件。当请求导致对象的 ACL 没有更改时，不会生成事件。

Amazon EventBridge 支持的事件类型

有关 Amazon S3 将发送到 Amazon EventBridge 的事件类型的列表，请参阅使用 EventBridge。

事件排序和重复事件

Amazon S3 事件通知设计为至少发送一次通知，但不能保证它们以与事件发生相同的顺序到达。在极少数情况下，Amazon S3 的重试机制可能会导致同一对象事件出现重复的 S3 事件通知。有关处理重复或无序事件的更多信息，请参阅 AWS 存储博客 上的 Manage event ordering and duplicate events with Amazon S3 Event Notifications。

使用 Amazon SQS、Amazon SNS 和 Lambda

启用通知是存储桶级操作。您可以在与该存储桶关联的通知子资源中存储通知配置信息。创建或更改存储桶通知配置后，通常必须等待 5 分钟才能使更改生效。首次启用通知时会发生 `s3:TestEvent`。您可以使用以下任意方法来管理通知配置：

- 使用 Amazon S3 控制台 – 控制台 UI 允许您在存储桶上设置通知配置，而无需编写任何代码。有关更多信息，请参阅 使用 Amazon S3 控制台启用和配置事件通知。
- 以编程方式使用 AWS SDK – 在内部，控制台和 SDK 都调用 Amazon S3 REST API 来管理与存储桶关联的通知子资源。有关使用 AWS SDK 的通知配置示例，请参阅 演练：为存储桶配置通知（SNS 主题或 SQS 队列）。

> **Note**
>
> 您也可以直接从代码中调用 Amazon S3 REST API。但是，这可能会比较繁琐，因为您必须编写代码对请求进行身份验证。

无论您使用哪种方法，Amazon S3 都将通知配置作为 XML 存储在与存储桶关联的通知子资源中。有关存储桶子资源的信息，请参阅 存储桶配置选项。

> **Note**
>
> 如果您由于已删除的目标而收到多个失败事件通知，则在尝试删除这些通知时，您可能会收到无法验证以下目标配置。您可以在 S3 控制台中通过同时删除所有失败的通知来解决此问题。

主题

授予将事件通知消息发布到目标的权限

您必须向 Amazon S3 主体授予调用相关 API 的必要权限，以将消息发布到 SNS 主题、SQS 队列或 Lambda 函数。这是为了让 Amazon S3 能够将事件通知消息发布到目标。

要解决向目标发布事件通知消息的问题，请参阅将 Amazon S3 事件通知发布到 Amazon Simple Notification Service 主题的疑难解答。

主题

授予调用 AWS Lambda 函数的权限

Amazon S3 通过调用 Lambda 函数并提供事件消息作为参数来将事件消息发布到 AWS Lambda。

使用 Amazon S3 控制台在 Amazon S3 存储桶上为 Lambda 函数配置事件通知时，控制台将在 Lambda 函数上设置必要的权限。这是为了让 Amazon S3 有权从存储桶调用函数。有关更多信息，请参阅 使用 Amazon S3 控制台启用和配置事件通知。

您还可以从 AWS Lambda 向 Amazon S3 授予调用 Lambda 函数的权限。有关更多信息，请参阅 AWS Lambda 开发人员指南中的教程：将 AWS Lambda 与 Amazon S3 结合使用。

授予将消息发布到 SNS 主题或 SQS 队列的权限

要授予 Amazon S3 权限以将消息发布到 SNS 主题或 SQS 队列，请将 AWS Identity and Access Management（IAM）策略附加到目标 SNS 主题或 SQS 队列。

有关如何将策略附加到 SNS 主题或 SQS 队列的示例，请参阅 演练：为存储桶配置通知（SNS 主题或 SQS 队列）。有关权限的更多信息，请参阅以下主题：

- Amazon Simple Notification Service 开发人员指南中的 Amazon SNS 访问控制示例案例
- Amazon Simple Queue Service 开发人员指南中的 Amazon SQS 中的 Identity and Access Management

目标 SNS 主题的 IAM 策略

以下是您附加到目标 SNS 主题的 AWS Identity and Access Management（IAM）策略示例。有关如何使用此策略为事件通知设置目标 Amazon SNS 主题的说明，请参阅 演练：为存储桶配置通知（SNS 主题或 SQS 队列）。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
```

```
            ],
            "Resource": "SNS-topic-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:bucket-name"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

目标 SQS 队列的 IAM 策略

以下是您附加到目标 SQS 队列的 IAM 策略示例。有关如何使用此策略为事件通知设置目标 Amazon SQS 队列的说明，请参阅 演练：为存储桶配置通知（SNS 主题或 SQS 队列）。

要使用此策略，您必须更新 Amazon SQS 队列 ARN、存储桶名称和存储桶拥有者的 AWS 账户 ID。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
            "Resource": "arn:aws:sqs:Region:account-id:queue-name",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
```

```
        }
    ]
}
```

对于 Amazon SNS 和 Amazon SQS IAM 策略，您可以在策略中指定 `StringLike` 条件而不是 `ArnLike` 条件。

使用 `ArnLike` 时，ARN 的分区、服务、账户 ID、资源类型和部分资源 ID 等部分必须与请求上下文中的 ARN 完全匹配。只有区域和资源路径允许部分匹配。

当使用 `StringLike` 而不是 `ArnLike` 时，无论哪个部分使用通配符，匹配都会忽略 ARN 结构并允许部分匹配。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素。

```
"Condition": {
  "StringLike": { "aws:SourceArn": "arn:aws:s3:*:*:bucket-name" }
  }
```

AWS KMS 密钥政策

如果 SQS 队列或 SNS 主题使用 AWS Key Management Service（AWS KMS）客户托管密钥进行加密，则必须向 Amazon S3 服务主体授予处理加密主题和/或队列的权限。要向 Amazon S3 服务主体授予权限，请将以下语句添加到客户托管密钥策略中。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

有关 AWS KMS 密钥策略的更多信息，请参阅 AWS Key Management Service 开发人员指南中的使用 AWS KMS 中的密钥策略。

有关将服务器端加密与适用于 Amazon SQS 和 Amazon SNS 的 AWS KMS 结合使用的更多信息，请参阅以下内容：

- mazon Simple Notification Service 开发人员指南中的密钥管理。
- Amazon Simple Queue Service 开发人员指南中的密钥管理。
- AWS 计算博客中的利用 AWS KMS 加密发布到 Amazon SNS 的消息。

使用 Amazon S3 控制台启用和配置事件通知

您可以启用特定的 Amazon S3 存储桶事件以在每次发生这些事件时向目标发送通知消息。本部分介绍了如何使用 Amazon S3 控制台启用事件通知。有关将事件通知与 AWS SDK 和 Amazon S3 REST API 结合使用的信息，请参阅 以编程方式配置事件通知。

先决条件：您必须先设置其中一个目标类型并配置权限，然后才能为存储桶启用事件通知。有关更多信息，请参阅受支持的事件目标 和授予将事件通知消息发布到目标的权限。

> **Note**
>
> 不支持将 Amazon Simple Queue Service FIFO（先进先出）队列作为 Amazon S3 事件通知目标。要向 Amazon SQS FIFO 队列发送 Amazon S3 事件的通知，您可以使用 Amazon EventBridge。有关更多信息，请参阅 启用 Amazon EventBridge。

主题

- 使用 Amazon S3 控制台启用 Amazon SNS、Amazon SQS 或 Lambda 通知

使用 Amazon S3 控制台启用 Amazon SNS、Amazon SQS 或 Lambda 通知

为 S3 存储桶启用和配置事件通知

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要为其启用事件的存储桶的名称。
3. 选择 Properties (属性)。

4. 导航到 Event Notifications (事件通知) 部分，然后选择 Create event notification (创建事件通知)。
5. 在 General configuration (常规配置) 部分中，为事件通知指定描述性事件名称。您还可以选择指定前缀和后缀，以将通知限制为键以指定字符结尾的对象。

   a. 为 Event name (事件名称) 输入描述。

      如果未输入名称，则将生成一个全局唯一标识符 (GUID) 并用作名称。

   b. （可选）要选择按前缀筛选事件通知，请输入 Prefix (前缀)。

      例如，可以设置前缀筛选器，使得仅在文件添加到特定文件夹（例如 `images/`）时，您才会收到通知。

   c. （可选）要选择按后缀筛选事件通知，请输入 Suffix (后缀)。

      有关更多信息，请参阅 使用对象键名筛选配置事件通知。
6. 在 Event types (事件类型) 部分中，选择要接收其通知的一个或多个事件类型。

   有关不同事件类型的列表，请参阅 SQS、SNS 和 Lambda 支持的事件类型。
7. 在 Destination (目标) 部分中，请选择事件通知目标。

   > **Note**
   >
   > 在发布事件通知之前，您必须向 Amazon S3 主体授予调用相关 API 的必要权限。这是为了使它可以将通知发布到 Lambda 函数、SNS 主题或 SQS 队列。

   a. 选择目标类型：Lambda Function (Lambda 函数)、SNS Topic (SNS 主题) 或 SQS Queue (SQS 队列)。

   b. 请选择目标类型后，从列表中选择函数、主题或队列。

   c. 或者，如果您希望指定 Amazon Resource Name (ARN)，请选择输入 ARN 并输入 ARN。

   有关更多信息，请参阅 受支持的事件目标。
8. 请选择 Save changes (保存更改)，Amazon S3 会向事件通知目标发送一条测试消息。

以编程方式配置事件通知

默认情况下，不为任何类型的事件启用通知。因此，通知子资源最初存储空配置。

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</NotificationConfiguration>
```

要为特定类型的事件启用通知，请将 XML 替换为适当的配置，该配置可标识您希望 Amazon S3 发布的事件类型和您希望将事件发布到的目标。对于每个目标，添加相应的 XML 配置。

将事件消息发布到 SQS 队列

要将 SQS 队列设置为一个或多个事件类型的通知目标，请添加 `QueueConfiguration`。

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>optional-id-string</Id>
    <Queue>sqs-queue-arn</Queue>
    <Event>event-type</Event>
    <Event>event-type</Event>
     ...
  </QueueConfiguration>
   ...
</NotificationConfiguration>
```

将事件消息发布到 SNS 主题

要将 SNS 主题设置为特定事件类型的通知目标，请添加 `TopicConfiguration`。

```
<NotificationConfiguration>
  <TopicConfiguration>
     <Id>optional-id-string</Id>
     <Topic>sns-topic-arn</Topic>
     <Event>event-type</Event>
     <Event>event-type</Event>
      ...
  </TopicConfiguration>
   ...
</NotificationConfiguration>
```

调用 AWS Lambda 函数并提供事件消息作为参数

要将 Lambda 函数设置为特定事件类型的通知目标，请添加`CloudFunctionConfiguration`。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
```

```
      <Id>optional-id-string</Id>
      <CloudFunction>cloud-function-arn</CloudFunction>
      <Event>event-type</Event>
      <Event>event-type</Event>
       ...
  </CloudFunctionConfiguration>
    ...
</NotificationConfiguration>
```

删除存储桶上配置的所有通知

要删除存储桶上配置的所有通知，请在通知子资源中保存一个空 <NotificationConfiguration/> 元素。

当 Amazon S3 检测到特定类型的事件时，它会发布包含事件信息的消息。有关更多信息，请参阅 事件消息结构。

有关配置事件通知的详细信息，请参阅以下主题：

- 演练：为存储桶配置通知（SNS 主题或 SQS 队列）.
- 使用对象键名筛选配置事件通知

演练：为存储桶配置通知（SNS 主题或 SQS 队列）

您可以使用 Amazon Simple Notification Service (Amazon SNS) 或 Amazon Simple Queue Service (Amazon SQS) 接收 Amazon S3 通知。在此演练中，您将使用 Amazon SNS 主题和 Amazon SQS 队列向存储桶添加通知配置。

> **Note**
>
> 不支持将 Amazon Simple Queue Service FIFO（先进先出）队列作为 Amazon S3 事件通知目标。要向 Amazon SQS FIFO 队列发送 Amazon S3 事件的通知，您可以使用 Amazon EventBridge。有关更多信息，请参阅 启用 Amazon EventBridge。

主题

- 演练摘要
- 第 1 步：创建 Amazon SQS 队列
- 第 2 步：创建 Amazon SNS 主题

- 步骤 3：将通知配置添加到存储桶
- 步骤 4：测试设置

演练摘要

此演练可以帮助您执行以下操作：

- 将 `s3:ObjectCreated:*` 类型的事件发布到 Amazon SQS 队列。
- 将 `s3:ReducedRedundancyLostObject` 类型的事件发布到 Amazon SNS 主题。

有关通知配置的信息，请参阅 使用 Amazon SQS、Amazon SNS 和 Lambda。

您可以使用控制台执行这些步骤，无需编写任何代码。此外，还提供了使用适用于 Java 和 .NET 的 AWS SDK 的代码示例，以帮助您通过编程方式添加通知配置。

该过程包括以下步骤：

1. 创建 Amazon SQS 队列。

   使用 Amazon SQS 控制台创建 SQS 队列。您可以通过编程方式访问 Amazon S3 发送到队列的所有消息。但对于此演练，您在控制台中验证通知消息。

   您可将访问策略附加到队列以授予 Amazon S3 发布消息的权限。
2. 创建 Amazon SNS 主题。

   使用 Amazon SNS 控制台创建 SNS 主题并订阅该主题。这样，发布到此项目的所有事件都传送给您。指定电子邮件作为通信协议。创建主题后，Amazon SNS 会发送电子邮件。您可使用电子邮件中的链接确认订阅主题。

   您可将访问策略附加到主题以授予 Amazon S3 发布消息的权限。
3. 将通知配置添加到存储桶。

第 1 步：创建 Amazon SQS 队列

按照以下步骤创建并订阅 Amazon Simple Queue Service (Amazon SQS) 队列。

1. 使用 Amazon SQS 控制台创建队列。有关说明，请参阅 Amazon Simple Queue Service 开发人员指南中的 Amazon SQS 入门。
2. 使用以下策略替换附加到队列的访问策略。

a. 在 Amazon SQS 控制台的 Queues (队列) 列表中，请选择队列名称。

b. 在 Access policy (访问策略) 选项卡上，请选择 Edit (编辑)。

c. 替换附加到队列的访问策略。在其中，提供您的 Amazon SQS ARN、源存储桶名称和存储桶拥有者账户 ID。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
            "Resource": "SQS-queue-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

d. 请选择保存。

3. （可选）如果 Amazon SQS 队列或 Amazon SNS 主题已通过 AWS Key Management Service (AWS KMS) 启用服务器端加密，可将以下策略添加到关联的对称加密客户托管式密钥。

   必须将此策略添加到客户托管密钥中，因为您无法为 Amazon SQS 或 Amazon SNS 修改 AWS 托管密钥。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
```

```
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

有关将适用于 Amazon SQS 和 Amazon SNS 的 SSE 与 AWS KMS 结合使用的更多信息，请参阅以下内容：

- mazon Simple Notification Service 开发人员指南中的密钥管理。
- Amazon Simple Queue Service 开发人员指南中的密钥管理。

4. 记录队列 ARN。

   您创建的 SQS 队列是您的 AWS 账户 中另一个资源。它有唯一的 Amazon Resource Name (ARN)。在下一步骤中，您需要用到此 ARN。ARN 的格式如下所示：

   ```
   arn:aws:sqs:aws-region:account-id:queue-name
   ```

第 2 步：创建 Amazon SNS 主题

按照以下步骤创建并订阅 Amazon SNS 主题。

1. 使用 Amazon SNS 控制台创建主题。有关说明，请参阅 Amazon Simple Notification Service 开发人员指南中的创建 Amazon SNS 主题。
2. 订阅至主题。对于此练习，请使用电子邮件作为通信协议。有关说明，请参阅 Amazon Simple Notification Service 开发人员指南中的订阅 Amazon SNS 主题。

   您将收到电子邮件要求您确认订阅该主题。确认订阅。

3. 使用以下策略替换附加到主题的访问策略。在其中，提供您的 SNS 主题 ARN、存储桶名称和存储桶拥有者账户 ID。

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "SNS-topic-ARN",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:bucket-name"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-account-id"
                }
            }
        }
    ]
}
```

4. 记录主题 ARN。

   您创建的 SNS 主题是您的 AWS 账户中的另一项资源，它有唯一的 ARN。在下一步骤中，您需要用到此 ARN。ARN 格式如下：

```
arn:aws:sns:aws-region:account-id:topic-name
```

步骤 3：将通知配置添加到存储桶

您可以使用 Amazon S3 控制台或以编程方式使用 AWS SDK 启用存储桶通知。请选择任何一个选项以配置存储桶通知。此部分提供使用适用于 Java 和 .NET 的 AWS SDK 的代码示例。

选项 A：使用控制台在存储桶上启用通知

使用 Amazon S3 控制台，添加请求 Amazon S3 执行以下操作的通知配置：

- 将所有对象创建事件类型的事件发布到 Amazon SQS 队列。
- 将 RRS 中的对象丢失类型的事件发布到 Amazon SNS 主题。

在您保存通知配置后，Amazon S3 会发布测试消息，您将通过电子邮件收到该消息。

有关说明，请参阅 使用 Amazon S3 控制台启用和配置事件通知。

选项 B：使用 AWS SDK 在存储桶上启用通知

.NET

下面的 C# 代码示例提供一个完整的代码列表，用于为存储桶添加通知配置。您必须更新该代码，提供您的存储桶名称和 SNS 主题 ARN。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class EnableNotificationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string snsTopic = "*** SNS topic ARN ***";
        private const string sqsQueue = "*** SQS topic ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            EnableNotificationAsync().Wait();
        }
```

```
        static async Task EnableNotificationAsync()
        {
            try
            {
               PutBucketNotificationRequest request = new
 PutBucketNotificationRequest
                {
                    BucketName = bucketName
                };

                TopicConfiguration c = new TopicConfiguration
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic
                };
                request.TopicConfigurations = new List<TopicConfiguration>();
                request.TopicConfigurations.Add(c);
                request.QueueConfigurations = new List<QueueConfiguration>();
                request.QueueConfigurations.Add(new QueueConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedPut },
                    Queue = sqsQueue
                });

                PutBucketNotificationResponse response = await
 client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' ",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown error encountered on server.
 Message:'{0}' ", e.Message);
            }
        }
    }
}
```

Java

以下示例说明如何将通知配置添加到存储桶。有关如何创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.EnumSet;

public class EnableNotificationOnABucket {

    public static void main(String[] args) throws IOException {
        String bucketName = "*** Bucket name ***";
        Regions clientRegion = Regions.DEFAULT_REGION;
        String snsTopicARN = "*** SNS Topic ARN ***";
        String sqsQueueARN = "*** SQS Queue ARN ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();
            BucketNotificationConfiguration notificationConfiguration = new
 BucketNotificationConfiguration();

            // Add an SNS topic notification.
            notificationConfiguration.addConfiguration("snsTopicConfig",
                    new TopicConfiguration(snsTopicARN,
 EnumSet.of(S3Event.ObjectCreated)));

            // Add an SQS queue notification.
            notificationConfiguration.addConfiguration("sqsQueueConfig",
                    new QueueConfiguration(sqsQueueARN,
 EnumSet.of(S3Event.ObjectCreated)));
```

```
            // Create the notification configuration request and set the bucket
 notification
            // configuration.
            SetBucketNotificationConfigurationRequest request = new
 SetBucketNotificationConfigurationRequest(
                    bucketName, notificationConfiguration);
            s3Client.setBucketNotificationConfiguration(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

步骤 4：测试设置

现在，您可以通过将对象上传到存储桶来测试设置，并在 Amazon SQS 控制台中验证事件通知。有关说明，请参阅 Amazon Simple Queue Service 开发人员指南“入门”部分中的接收消息。

使用对象键名筛选配置事件通知

配置 Amazon S3 事件通知时，您必须指定哪些受支持的 Amazon S3 事件类型会导致 Amazon S3 发送通知。如果 S3 存储桶中发生了您未指定的事件类型，Amazon S3 不会发送通知。

您可以将通知配置为按对象的键名的前缀和后缀进行筛选。例如，您可以设置一个配置，以便仅在带有 ".jpg" 文件扩展名的图像文件添加到存储桶时收到通知。或者，您也可以设置一个配置，该配置仅在将带有前缀“images/”的对象添加到存储桶时将通知发送到 Amazon SNS 主题，同时将同一存储桶中带有前缀“logs/”的对象的通知发送到 AWS Lambda 函数。

> **Note**
>
> 通配符（"*"）不能在筛选条件中用作前缀或后缀。如果您的前缀或后缀包含空格，则必须将其替换为“+”字符。如果您在前缀或后缀的值中使用任何其他特殊字符，您必须以 URL 编码（百

分比编码）格式输入它们。有关在事件通知的前缀或后缀中使用时必须转换为 URL 编码格式的特殊字符的完整列表，请参阅安全字符。

您可以在 Amazon S3 控制台中设置使用对象密钥名称过滤的通知配置。您可以直接通过 AWS SDK 或 REST API 使用 Amazon S3 API 来实现该操作。有关使用控制台 UI 在存储桶上设置通知配置的信息，请参阅使用 Amazon S3 控制台启用和配置事件通知。

Amazon S3 在与存储桶关联的通知子资源中以 XML 形式存储通知配置，如 使用 Amazon SQS、Amazon SNS 和 Lambda 中所述。您使用 `Filter` XML 结构定义要按对象键名的前缀和后缀进行筛选的通知的规则。有关 `Filter` XML 结构的信息，请参阅 Amazon Simple Storage Service API 参考中的 PUT Bucket 通知。

使用 `Filter` 的通知配置无法定义采用重叠前缀、重叠后缀或前缀和后缀重叠的筛选规则。以下部分提供了使用对象键名称筛选的有效通知配置的示例。它们还包含由于前缀和后缀重叠而无效的通知配置的示例。

主题

- 采用对象键名筛选的有效通知配置的示例
- 采用无效前缀和后缀重叠的通知配置的示例

采用对象键名筛选的有效通知配置的示例

以下通知配置包含用于标识 Amazon S3 的 Amazon SQS 队列以发布 `s3:ObjectCreated:Put` 类型的事件的队列配置。每当具有前缀 `images/` 和后缀 `jpg` 的对象被放置（PUT）到存储桶时，就会发布这些事件。

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>images/</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>jpg</Value>
```

```
              </FilterRule>
          </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:s3notificationqueue</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

以下通知配置有多个非重叠前缀。该配置作出以下定义：images/ 文件夹中针对 PUT 请求的通知进入队列 A，logs/ 文件夹中针对 PUT 请求的通知进入队列 B。

```
<NotificationConfiguration>
  <QueueConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images/</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-A</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
  <QueueConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>logs/</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-B</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

以下通知配置有多个非重叠后缀。该配置作出以下定义：所有新添加到存储桶的 .jpg 映像都由 Lambda cloud-function-A 处理，所有新添加的 .png 映像都由云函数 B 处理。.png 和 .jpg 后缀不

重叠，即使它们有相同的最后一个字母也是如此。如果某个给定的字符串能够以这两个后缀结尾，则将它们视为重叠。一个字符串无法同时以 .png 和 .jpg 结尾，因此示例配置中的后缀不是重叠后缀。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

使用 Filter 的通知配置不能使用重叠前缀为相同的事件类型定义筛选规则。只有在与后缀不重叠一起使用的重叠前缀时，他们才能这样做。以下示例配置显示了如何将使用通用前缀和非重叠后缀创建的对象传递到其他目标。

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
```

```
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

采用无效前缀和后缀重叠的通知配置的示例

大多数情况下，使用 `Filter` 的通知配置无法使用重叠前缀、重叠后缀或前缀和后缀的重叠组合为相同的事件类型定义筛选规则。只要后缀不重叠，您就可以采用重叠前缀。有关示例，请参阅 使用对象键名筛选配置事件通知。

您可以将重叠对象键名筛选用于不同的事件类型。例如，您可以创建一个通知配置，该配置将前缀 image/ 用于 ObjectCreated:Put 事件类型并将前缀 image/ 用于 ObjectRemoved:* 事件类型。

在使用 Amazon S3 控制台或 API 时，如果您尝试保存具有对相同的活动类型无效的重叠名称筛选的通知配置，则会出现错误。此部分显示了因重叠名称筛选而无效的通知配置的示例。

任何现有通知配置规则都被假定为具有与任何其他前缀和后缀分别匹配的默认前缀和后缀。由于具有重叠前缀，以下通知配置无效。具体来说，根前缀与任何其他前缀重叠。（如果在此示例中使用后缀而不是前缀，则会发生同样的情况。根后缀与任何其他后缀重叠。）

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-two</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

由于具有重叠后缀，以下通知配置无效。如果某个给定的字符串能够以这两个后缀结尾，则将它们视为重叠。字符串可以用 jpg 和 pg 结尾。因此，后缀会重叠。（前缀也是如此。如果给定字符串能够以两个前缀开头，则认为这两个前缀是重叠的。

```
 <NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
```

```
                    <Value>jpg</Value>
                </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>pg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration
```

由于具有重叠前缀和后缀，以下通知配置无效。

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
```

```
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>jpg</Value>
                </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

事件消息结构

Amazon S3 为发布事件而发送的通知消息采用 JSON 格式。

有关配置事件通知的一般概述和说明，请参阅 Amazon S3 事件通知。

此示例说明了事件通知 JSON 结构的版本 2.2。Amazon S3 使用了这个事件结构的版本 2.1、2.2 和 2.3。Amazon S3 使用版本 2.2 进行跨区域复制事件通知。它将 2.3 版本用于 S3 生命周期、S3 Intelligent-Tiering、对象 ACL、对象标记和对象恢复删除事件。这些版本包含特定于这些操作的额外信息。版本 2.2 和版本 2.3 与版本 2.1 兼容，Amazon S3 当前将该版本用于所有其他事件类型。

```
{
   "Records":[
      {
         "eventVersion":"2.2",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"The time, in ISO-8601 format, for example,
 1970-01-01T00:00:00.000Z, when Amazon S3 finished processing the request",
         "eventName":"event-type",
         "userIdentity":{
            "principalId":"Amazon-customer-ID-of-the-user-who-caused-the-event"
         },
         "requestParameters":{
            "sourceIPAddress":"ip-address-where-request-came-from"
         },
         "responseElements":{
            "x-amz-request-id":"Amazon S3 generated request ID",
            "x-amz-id-2":"Amazon S3 host that processed the request"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"ID found in the bucket notification configuration",
            "bucket":{
```

```
                "name":"bucket-name",
                "ownerIdentity":{
                   "principalId":"Amazon-customer-ID-of-the-bucket-owner"
                },
                "arn":"bucket-ARN"
             },
             "object":{
                "key":"object-key",
                "size":"object-size in bytes",
                "eTag":"object eTag",
                "versionId":"object version if bucket is versioning-enabled, otherwise
 null",
                "sequencer": "a string representation of a hexadecimal value used to
 determine event sequence, only used with PUTs and DELETEs"
             }
          },
          "glacierEventData": {
             "restoreEventData": {
                "lifecycleRestorationExpiryTime": "The time, in ISO-8601 format, for
 example, 1970-01-01T00:00:00.000Z, of Restore Expiry",
                "lifecycleRestoreStorageClass": "Source storage class for restore"
             }
          }
       }
    ]
}
```

注意，以下是关于事件消息结构：

- eventVersion 键值包含 <major>.<minor> 格式的主要版本和次要版本。

  如果 Amazon S3 对不向后兼容的事件结构进行更改，则主要版本将递增。这包括删除已存在的 JSON 字段或更改字段内容的表示方式（例如，日期格式）。

  如果 Amazon S3 向事件结构添加新字段，则次要版本将递增。如果为某些或所有现有事件提供了新信息，则可能会发生这种情况 如果仅提供有关新引入的事件类型的新信息，则可能会发生这种情况。应用程序应忽略新字段，以保持与事件结构的新次要版本的前向兼容。

  如果引入了新的事件类型但事件的结构未经修改，则事件版本不会更改。

  为确保您的应用程序可以正确解析事件结构，我们建议您对主要版本号进行相等比较。为了确保您的应用程序所期望的字段存在，我们还建议对次要版本进行大于或等于比较。

- eventName 引用事件通知类型列表，但不包含 s3: 前缀。
- 如果您希望通过联系 AWS Support 来跟踪请求，则 responseElements 键值很有用。x-amz-request-id 和 x-amz-id-2 都可帮助 Amazon S3 跟踪单个请求。这些值与 Amazon S3 为响应您启动事件的请求而返回的值相同。这是因为它们可以用于将事件与请求匹配。
- s3 键提供事件中涉及的存储桶和对象的相关信息。对象键名称值进行了 URL 编码。例如，“red flower.jpg”变成了“red+flower.jpg”（Amazon S3 返回“application/x-www-form-urlencoded”作为响应中的内容类型）。
- sequencer 键提供了确定事件顺序的方法。无法保证事件通知按事件发生的顺序到达。然而，来自创建对象 (PUTs) 和删除对象事件的通知将包含 sequencer。它可用于确定给定对象密钥的事件的顺序。

  如果将来自同一对象键上的两个事件通知的 sequencer 字符串进行比较，就会发现 sequencer 十六进制值较大的事件通知是后发生的事件。如果您正在使用事件通知来维护 Amazon S3 对象的单独数据库或索引，我们建议您在处理每个事件通知时比较和存储 sequencer 值。

  请注意以下几点：

  - 您不能使用 sequencer 确定不同对象键上的事件的顺序。
  - 排序器可以有不同的长度。因此，要比较这些值，您必须先使用零填补较短的值的右侧，然后进行字母表顺序比较。
- glacierEventData 键只对 s3:ObjectRestore:Completed 事件可见。
- restoreEventData 键包含与您的还原请求相关的属性。
- replicationEventData 键只对复制事件可见。
- intelligentTieringEventData 密钥只对 S3 Intelligent-Tiering 可见。
- lifecycleEventData 密钥只对 S3 生命周期转换事件可见。

示例消息

以下是 Amazon S3 事件通知消息的示例。

Amazon S3 测试消息

当您在存储桶上配置事件通知时，Amazon S3 会发送以下测试消息。

```
{
   "Service":"Amazon S3",
```

```
   "Event":"s3:TestEvent",
   "Time":"2014-10-13T15:57:02.089Z",
   "Bucket":"bucketname",
   "RequestId":"5582815E1AEA5ADF",
   "HostId":"8cLeGAmw098X5cv4Zkwcmo8vvZa3eH3eKxsPzbB9wrR+YstdA6Knx4Ip8EXAMPLE"
}
```

使用 PUT 请求创建对象时的示例消息

以下消息是 Amazon S3 发送以发布 `s3:ObjectCreated:Put` 事件的消息示例。

```
{
   "Records":[
      {
         "eventVersion":"2.1",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"ObjectCreated:Put",
         "userIdentity":{
            "principalId":"AIDAJDPLRKLG7UEXAMPLE"
         },
         "requestParameters":{
            "sourceIPAddress":"127.0.0.1"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"mybucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::mybucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
```

```
                "versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko",
                "sequencer":"0055AED6DCD90281E5"
            }
        }
    }
  ]
}
```

有关每个 IAM 标识前缀（如 AIDA、AROA、AGPA 等）的定义，请参阅 IAM 用户指南中的 IAM 标识符。

使用 EventBridge

无论何时在存储桶中发生某些事件，Amazon S3 都可以将事件发送到 Amazon EventBridge。与其他目的地不同，您无需选择要提供的活动类型。启用 EventBridge 后，以下所有事件都将发送到 EventBridge。您可以使用 EventBridge 规则将事件路由到其他目标。以下列出了 Amazon S3 发送给 EventBridge 的事件。

事件类型	说明
Object Created（已创建对象）	创建了一个对象。 事件消息结构中的原因字段表示使用哪个 S3 API 创建对象：PutObject、POST 对象、CopyObject 还是 CompleteMultipartUpload。
已删除对象 (DeleteObject) 已删除对象（生命周期过期）	已删除对象。 使用 S3 API 调用 DeleteObject 时，原因字段将设置为 deleteObject。当 S3 生命周期过期规则删除对象时，原因字段将设置为生命周期过期。有关更多信息，请参阅 即将过期的对象。 当删除未受版本控制的对象或永久删除受版本控制的对象时，删除类型字段将设置为永久删除。为受版本控制的对象创建删除标记时，删除类型字段将设置为删除标记已创建。有关更多信息，请参阅 从启用了版本控制的存储桶中删除对象版本。

事件类型	说明
Object Restore Initiated（已启动对象还原）	从 S3 Glacier 或 S3 Glacier Deep Archive 存储类或 S3 Intelligent-Tiering 归档访问层或深度归档访问层启动对象还原。有关更多信息，请参阅 使用归档的对象。
Object Restore Completed（已完成对象还原）	对象还原已完成。
Object Restore Expired（对象还原已过期）	从 S3 Glacier 或 S3 Glacier Deep Archive 还原的对象的临时副本过去并被删除。
Object Storage Class Changed（对象存储类已更改）	对象已过渡到不同的存储类。有关更多信息，请参阅 使用 Amazon S3 生命周期转换对象。
Object Access Tier Changed（对象访问层已更改）	对象已转换为 S3 Intelligent-Tiering 归档访问层或深度归档访问层。有关更多信息，请参阅 Amazon S3 Intelligent-Tiering。
Object ACL Updated（已更新对象 ACL）	对象的访问控制列表 (ACL) 是使用 putObjecTACL 设置的。当请求导致对象的 ACL 没有更改时，不会生成事件。有关更多信息，请参阅 访问控制列表 (ACL) 概述。
Object Tags Added（已添加对象标签）	使用 PutObjectTagging 向对象添加了一组标签。有关更多信息，请参阅 使用标签对存储进行分类。
Object Tags Deleted（已删除对象标签）	使用 DeleteObjectTagging 从对象中移除了所有标签。有关更多信息，请参阅 使用标签对存储进行分类。

> **Note**
>
> 有关 Amazon S3 事件类型如何映射到 EventBridge 事件类型的更多信息，请参阅 Amazon EventBridge 映射和故障排除。

您可以将 Amazon S3 事件通知与 EventBridge 结合使用，编写规则，以便在存储桶中发生事件时采取行动。例如，可获得向您发送的电子邮件通知。有关更多信息，请参阅 Amazon EventBridge 用户指南中的什么是 EventBridge。

有关可以使用 EventBridge API 进行交互的操作和数据类型的更多信息，请参阅《Amazon EventBridge API 参考》中的 Amazon EventBridge API Reference。

有关定价的更多信息，请参阅 Amazon EventBridge pricing（Amazon EventBridge 定价）。

主题

Amazon EventBridge 权限

Amazon S3 不需要任何其他权限即可向 Amazon EventBridge 传输事件。

启用 Amazon EventBridge

您可以通过 S3 控制台、AWS Command Line Interface (AWS CLI) 或 Amazon S3 REST API 启用 Amazon EventBridge。

使用 S3 控制台

在 S3 控制台中启用 EventBridge 事件传输。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要为其启用事件的存储桶的名称。
3. 选择属性。
4. 导航到 Event Notifications（事件通知）章节，然后找到 Amazon EventBridge 小节。请选择编辑。
5. 在 Send notifications to Amazon EventBridge for all events in this bucket（为此存储桶中的所有事件向 Amazon EventBridge 发送通知）下方，请选择 On（打开）。

> **Note**
>
> 在启用 EventBridge 后，所做的更改需要大约五分钟才能生效。

使用 AWS CLI

以下示例为启用了 Amazon EventBridge 的存储桶 amzn-s3-demo-bucket1 创建了一个存储桶通知配置。

```
aws s3api put-bucket-notification-configuration --bucket amzn-s3-demo-bucket1 --
notification-configuration='{ "EventBridgeConfiguration": {} }'
```

使用 REST API

您可以通过调用 Amazon S3 REST API 以编程方式在存储桶上启用 Amazon EventBridge。有关更多信息，请参阅 Amazon Simple Storage Service API Reference 中的 PutBucketNotificationConfiguration。

下面的例子展示了在启用 Amazon EventBridge 的情况下，用来创建存储桶通知配置的 XML。

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <EventBridgeConfiguration>
  </EventBridgeConfiguration>
</NotificationConfiguration>
```

创建 EventBridge 规则

启用后，您可以为某些任务创建 Amazon EventBridge 规则。例如，您可以在创建对象时发送电子邮件通知。有关完整的教程，请参阅 Amazon EventBridge User Guide（Amazon EventBridge 用户指南）中的 Tutorial: Send a notification when an Amazon S3 object is created（教程：创建 Amazon S3 对象时发送通知）。

EventBridge 事件消息结构

Amazon S3 为发布事件而发送的通知消息采用 JSON 格式。当 Amazon S3 将事件发送到 Amazon EventBridge 时，会显示以下字段。

- version（版本）— 目前所有活动均为 0（零）。
- id — 为每个事件生成版本 4 UUID。
- detail-type— 正在发送的事件类型。有关事件类型的列表，请参阅 使用 EventBridge。
- source（源）— 识别生成事件的服务。
- account（账户）- 存储桶拥有者的 12 位 AWS 账户 ID。
- time（时间）- 事件发生的时间。

- region（区域）—识别存储桶的 AWS 区域。
- resource（资源）- 包含存储桶的 Amazon 资源名称（ARN）的 JSON 数组。
- detail（详细信息）—包含关于事件信息的 JSON 对象。有关此字段中可包含的内容的更多信息，请参阅事件消息详细信息。

事件消息结构示例

以下是一些可以发送到 Amazon EventBridge 的 Amazon S3 事件通知消息的示例。

已创建对象

```
{
  "version": "0",
  "id": "17793124-05d4-b198-2fde-7ededc63b103",
  "detail-type": "Object Created",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "IYV3p45BT0ac8hjHg1houSdS1a.Mro8e",
      "sequencer": "617f08299329d189"
    },
    "request-id": "N4N7GDK58NMKJ12R",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "PutObject"
  }
}
```

已删除对象（使用 DeleteObject）

```
{
  "version": "0",
  "id": "2ee9cc15-d022-99ea-1fb8-1b1bac4850f9",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "1QW9g1Z99LUNbvaaYVpW9xDlOLU.qxgF",
      "sequencer": "617f0837b476e463"
    },
    "request-id": "0BH729840619AG5K",
    "requester": "123456789012",
    "source-ip-address": "1.2.3.4",
    "reason": "DeleteObject",
    "deletion-type": "Delete Marker Created"
  }
}
```

已删除对象（使用生命周期过期）

```
{
  "version": "0",
  "id": "ad1de317-e409-eba2-9552-30113f8d88e3",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
```

```
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "mtB0cV.jejK63XkRNceanNMC.qXPWLeK",
      "sequencer": "617b398000000000"
    },
    "request-id": "20EB74C14654DC47",
    "requester": "s3.amazonaws.com",
    "reason": "Lifecycle Expiration",
    "deletion-type": "Delete Marker Created"
  }
}
```

已完成对象还原

```
{
  "version": "0",
  "id": "6924de0d-13e2-6bbf-c0c1-b903b753565e",
  "detail-type": "Object Restore Completed",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
```

```
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "KKsjUC1.6gIjqtvhfg5AdMI0eCePIiT3"
    },
    "request-id": "189F19CB7FB1B6A4",
    "requester": "s3.amazonaws.com",
    "restore-expiry-time": "2021-11-13T00:00:00Z",
    "source-storage-class": "GLACIER"
  }
}
```

事件消息详细信息

详细信息字段包含一个 JSON 对象，其中包含事件相关信息。详细信息字段中可能存在以下字段。

- version（版本）— 目前所有活动均为 0（零）。
- bucket（存储桶）— 有关事件中涉及的 Amazon S3 存储桶的相关信息。
- object（对象）— 有关事件中涉及的 Amazon S3 对象的相关信息。
- request-id — S3 响应中的请求 ID。
- requester（请求者）—请求者的 AWS 账户 ID 或 AWS 服务主体。
- source-ip-address — S3 请求的源 IP 地址。仅用于 S3 请求触发的事件。
- 原因 — 对于 Object Created（已创建对象）事件，S3 API 用来创建对象：PutObject、POST 对象、CopyObject 或 CompleteMultipartUpload。对于 Object Deleted（已删除对象），当对象由 S3 API 调用删除时，这将设为 DeleteObject，或者当对象由 S3 生命周期过期规则删除时，这将设为 Lifecycle Expiration。有关更多信息，请参阅 即将过期的对象。
- deletion-type — 对于 Object Deleted（已删除对象），当删除未受版本控制的对象或永久删除受版本控制的对象时，删除类型字段将设置为永久删除。为受版本控制的对象创建删除标记时，此将设为 Delete Marker Created（已创建删除标记）。有关更多信息，请参阅 从启用了版本控制的存储桶中删除对象版本。

  > **Note**
  >
  > 某些对象属性（例如 `etag` 和 `size`）仅在创建删除标记时才会出现。

- restore-expiry-time — 对于 Object Restore Completed（已完成对象还原）事件，该对象的临时副本将从 S3 中删除的时间。有关更多信息，请参阅 使用归档的对象。

- source-storage-class — 对于 Object Restore Initiated（已启动对象还原）和 Object Restore Completed（已完成对象还原）事件，即正在恢复的对象的存储类。有关更多信息，请参阅 使用归档的对象。
- destination-storage-class — 对于 Object Storage Class Changed（已更改对象存储类）事件，对象的新存储类。有关更多信息，请参阅 使用 Amazon S3 生命周期转换对象。
- destination-access-tier — 对于 Object Access Tier Changed（已更改对象访问层）事件，对象的新访问层。有关更多信息，请参阅 Amazon S3 Intelligent-Tiering。

Amazon EventBridge 映射和故障排除

下表介绍了 Amazon S3 事件类型如何映射到 Amazon EventBridge 事件类型。

S3 事件类型	Amazon EventBridge 详细信息类型
ObjectCreated:Put ObjectCreated:Post ObjectCreated:Copy ObjectCreated:CompleteMultipartUpload	已创建对象
ObjectRemoved：删除 ObjectRemoved:DeleteMarkerCreated LifecycleExpiration：删除 LifecycleExpiration:DeleteMarkerCreated	已删除对象
ObjectRestore:Post	已启动对象还原
ObjectRestore:Completed	已完成对象还原
ObjectRestore：删除	对象还原已过期

S3 事件类型	Amazon EventBridge 详细信息类型
LifecycleTransition	已更改对象存储类
IntelligentTiering	已更改对象访问层
ObjectTagging:Put	已添加对象标签
ObjectTagging:Delete	已删除对象标签
ObjectAcl:Put	已更新对象 ACL

Amazon EventBridge 故障排除

有关 EventBridge 故障排除的信息，请参阅 Amazon EventBridge User Guide（Amazon EventBridge 用户指南）中的 Troubleshooting Amazon EventBridge（对 Amazon EventBridge 进行故障排除）。

使用分析和见解

您可以使用 Amazon S3 中的分析和见解来了解、分析和优化存储使用情况。有关更多信息，请参阅以下主题。

主题

Amazon S3 分析 – 存储类分析

通过使用 Amazon S3 分析存储类分析，您可以分析存储访问模式以帮助您决定何时将正确的数据转换为正确的存储类。这个新的 Amazon S3 分析功能可观察数据访问模式以帮助您确定何时将不常访问的 STANDARD 存储转换为 STANDARD_IA（IA，适用于不频繁访问）存储类。有关存储类的更多信息，请参阅 使用 Amazon S3 存储类。

在存储类分析在一段时间内观察到一组筛选出的数据的不常访问模式后，您可以使用分析结果来帮助改进您的生命周期配置。您可以将存储类分析配置为分析存储桶中的所有对象。或者，也可以配置筛选条件以按通用前缀 (即，名称以通用字符串开头的对象)、对象标签或前缀和标签的组合对对象进行分组以便进行分析。您很有可能会发现，按对象组进行筛选是从存储类分析获益的最佳方式。

> **Important**
>
> 存储类分析仅针对标准类到标准 IA 类提供建议。

每个存储桶可具有多个存储类分析筛选条件 (最多 1,000 个)，并且将收到针对每个筛选条件的单独分析。利用多个筛选配置，您可以分析特定对象组，以改进将对象转换为 STANDARD_IA 的生命周期配置。

存储类分析提供了 Amazon S3 控制台中的存储使用率可视化项，这些可视化项每天将进行更新。您也可以将此类日常使用数据导出到 S3 存储桶，并在电子表格应用程序中或使用商业智能工具（如 Amazon QuickSight）查看它们。

存储类分析会产生相关的成本。有关定价信息，请参阅管理和复制 Amazon S3 定价。

主题

如何设置存储类分析？

通过配置要分析的对象数据来设置存储类分析。您可将存储类分析配置为执行以下操作：

- 分析存储桶的全部内容。

  您将收到针对存储桶中所有对象的分析。
- 分析按前缀和标签分组的对象。

  您可以配置按前缀、对象标签或前缀和标签的组合对对象进行分组以进行分析的筛选条件。您将收到针对配置的每个筛选条件的单独分析。每个存储桶可具有多个筛选配置（最多 1000 个）。
- 导出分析数据。

  当您为存储桶或筛选条件配置存储类分析时，可以选择每天将分析数据导出到一个文件。当天的分析将添加到该文件以形成针对配置的筛选条件的历史分析日志。该文件会在所选目标上每天进行更新。选择要导出的数据时，指定写入文件的目标存储桶和可选目标前缀。

您可以使用 Amazon S3 控制台、REST API、AWS CLI 或 AWS SDK 配置存储类分析。

- 有关如何在 Amazon S3 控制台中配置存储类分析的信息，请参阅 配置存储类分析。
- 要使用 Amazon S3 API，可使用 PutBucketAnalyticsConfiguration REST API 或者 AWS CLI 或 AWS SDK 中的等效项。

如何使用存储类分析？

您使用存储类分析观察数据访问模式一段时间来收集信息，以帮助您改进 STANDARD_IA 存储的生命周期管理。配置一个筛选条件后，您首先会在 Amazon S3 控制台中看到 24 到 48 小时内基于该筛选条件的数据分析。但是，存储类分析会观察筛选的数据集的访问模式 30 天或更长时间，以便在提供结果前收集用于分析的信息。分析会在初始结果后继续运行，并在访问模式发生更改时更新结果。

首次配置筛选条件时，Amazon S3 控制台可能需要一些时间来分析您的数据。

存储类分析观察筛选的对象数据集的访问模式 30 天或更长时间，以便收集足量信息来进行分析。在存储类分析收集足量信息后，您将在 Amazon S3 控制台中看到一条表示分析已完成的消息。

对不常访问的对象执行分析时，存储类分析基于期限查看已分组的筛选的对象集，因为这些对象已上传到 Amazon S3。存储类分析通过查看筛选的数据集的以下因素来确定是否不常访问期限组：

- STANDARD 存储类中大于 128 KB 的对象。
- 每个期限组内的平均总存储量。
- 每个期限组传出的平均字节数 (非频率)。
- 分析导出数据仅包含与存储类分析有关的数据请求。这可能导致相对于存储指标所示或您自己内部系统的跟踪结果，请求数量以及上传和请求总字节数存在差异。
- 失败的 GET 和 PUT 请求不计入分析。但是，您会在存储指标中看到失败的请求。

我检索了多少我的存储空间？

Amazon S3 控制台用图表表示观察期间检索到的筛选的数据集中的存储空间量。

我检索了多少百分比的我的存储空间？

Amazon S3 控制台也用图表表示观察期间检索到的筛选的数据集中的存储空间百分比。

如本主题中先前所述，当您对不常访问的对象执行分析时，存储类分析会基于期限查看已分组的筛选的对象集，因为这些对象已上传到 Amazon S3。存储类分析使用以下预定义的对象期限组：

- 不超过 15 天的 Amazon S3 对象
- 15-29 天的 Amazon S3 对象
- 30-44 天的 Amazon S3 对象
- 45-59 天的 Amazon S3 对象
- 60-74 天的 Amazon S3 对象
- 75-89 天的 Amazon S3 对象
- 90-119 天的 Amazon S3 对象
- 120-149 天的 Amazon S3 对象
- 150-179 天的 Amazon S3 对象
- 180-364 天的 Amazon S3 对象
- 365-729 天的 Amazon S3 对象

- 730 天及更长时间的 Amazon S3 对象

通常大约需要 30 天来观察访问模式以收集足够信息来获得分析结果。根据您的数据的独特访问模式，这可能需要 30 天以上的时间。但在配置一个筛选条件后，您首先会在 Amazon S3 控制台中看到 24 到 48 小时内基于该筛选条件的数据分析。您可以查看 Amazon S3 控制台中每天按对象期限组划分的对象访问的分析。

我的多少存储空间是不常访问的？

Amazon S3 控制台显示按预定义的对象期限组分组的访问模式。显示的 Frequently accessed (经常访问) 或 Infrequently accessed (不经常访问) 文本旨在以直观的方式帮助您完成生命周期创建过程。

如何导出存储类分析数据？

您可以选择使存储类分析将分析报告导出到逗号分隔值 (CSV) 平面文件。报告每天都会更新且基于您配置的对象期限组筛选条件。使用 Amazon S3 控制台时，您可以在创建筛选条件时选择导出报告选项。选择要导出的数据时，指定写入文件的目标存储桶和可选目标前缀。您可以将数据导出到不同账户中的目标存储桶。目标存储桶必须位于与您配置为进行分析的存储桶相同的区域中。

您必须在目标存储桶上创建存储桶策略以向 Amazon S3 授予验证哪些 AWS 账户 拥有存储桶以及将对象写入定义位置的存储桶中的权限。有关策略示例，请参阅 向 S3 清单和 S3 分析功能授予权限。

在配置存储类分析报告后，您将在 24 小时后开始获得每日导出报告。之后，Amazon S3 会继续监控并提供每日导出。

您可以在电子表格应用程序中打开此 CSV 文件，或将此文件导入其他应用程序中，如 Amazon QuickSight。有关将 Amazon S3 文件用于 Amazon QuickSight 的信息，请参阅《Amazon QuickSight 用户指南》中的使用 Amazon S3 文件创建数据集。

导出的文件中的数据在对象期限组内按日期进行排序，如以下示例所示。如果存储类为 STANDARD，则行还包含 ObjectAgeForSIATransition 和 RecommendedObjectAgeForSIATransition 列的数据。

Date	ConfigId	Filter	StorageClass	ObjectAge	ObjectCount	DataUploaded_MB	Storage_MB	DataRetrieved_MB	GetRequestCount	CumulativeAccessRatio	ObjectAgeForSIATransition	RecommendedObjectAgeForSIATransition
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/2/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/5/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		

报告结束时，给出的对象期限组为 ALL。ALL 行包含当天所有使用期限组的累计总数，包括小于 128 KB 的对象。

8/24/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
9/3/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.02426125	015-029
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.03545875	015-029
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.0209529	015-029
9/4/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.02304819	015-029
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0.03073092	015-029
8/20/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3	0.4599		0	000-014

下一部分介绍报告中使用的列。

导出的文件布局

下表描述导出的文件的布局。

配置存储类分析

通过使用 Amazon S3 分析存储类分析工具，您可以分析存储访问模式以帮助您决定何时将正确的数据转换为正确的存储类。存储类分析发现数据访问模式以帮助您确定何时将不常访问的 STANDARD 存储转换为 STANDARD_IA (IA，适用于不常访问) 存储类。有关 STANDARD_IA 的更多信息，请参阅 Amazon S3 常见问题和 使用 Amazon S3 存储类。

通过配置要分析的对象数据来设置存储类分析。您可将存储类分析配置为执行以下操作：

- 分析存储桶的全部内容。

  您将收到针对存储桶中所有对象的分析。

- 分析按前缀和标签分组的对象。

  您可以配置按前缀、对象标签或前缀和标签的组合对对象进行分组以进行分析的筛选条件。您将收到针对配置的每个筛选条件的单独分析。每个存储桶可具有多个筛选配置（最多 1000 个）。

- 导出分析数据。

  当您为存储桶或筛选条件配置存储类分析时，可以选择每天将分析数据导出到一个文件。当天的分析将添加到该文件以形成针对配置的筛选条件的历史分析日志。该文件会在所选目标上每天进行更新。选择要导出的数据时，指定写入文件的目标存储桶和可选目标前缀。

您可以使用 Amazon S3 控制台、REST API、AWS CLI 或 AWS SDK 配置存储类分析。

> **Important**
>
> 存储类分析不提供转换到 ONEZONE_IA 或 S3 Glacier Flexible Retrieval 存储类的建议。

如果要配置存储类分析以将结果导出为 .csv 文件，并且目标存储桶使用带有 AWS KMS key 的原定设置存储桶加密，则必须更新 AWS KMS 密钥策略以授予 Amazon S3 加密 .csv 文件的权限。有关说明，请参阅 向 Amazon S3 授予权限以使用客户自主管理型密钥进行加密。

有关分析的更多信息，请参阅 Amazon S3 分析 – 存储类分析。

使用 S3 控制台

配置存储类分析

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets (存储桶) 列表中，请选择要为其配置存储类分析的存储桶的名称。
3. 请选择 Metrics (指标) 选项卡。
4. 在 Storage Class Analysis（存储类分析）下，请选择 Create analytics configuration（创建分析配置）。
5. 为筛选器键入名称。如果要分析整个存储桶，请将 Prefix（前缀）字段保留为空。
6. 在 Prefix（前缀）字段中，键入要分析的对象的前缀文本。
7. 要添加标签，请选择添加标签。输入标签的键和值。您可以输入一个前缀和多个标签。
8. （可选）您可以选择导出 CSV 下的启用，以将分析报告导出到逗号分隔值（.csv）平面文件。请选择可将文件存储到的目标存储桶。您可以键入目标存储桶的前缀。目标存储桶必须位于与您为其设置分析的存储桶相同的 AWS 区域 中。目标存储桶可处于不同的 AWS 账户 中。

   如果 .csv 文件的目标存储桶使用 KMS 密钥的原定设置存储桶加密，则必须更新 AWS KMS 密钥策略以授予 Amazon S3 加密 .csv 文件的权限。有关说明，请参阅 向 Amazon S3 授予权限以使用客户自主管理型密钥进行加密。
9. 请选择 Create Configuration (创建配置)。

Amazon S3 在授予 Amazon S3 写入权限的目标存储桶上创建存储桶策略。此操作允许它将导出数据写入存储桶。

如果在您尝试创建存储桶策略出现错误，则将为您提供相关修复说明。例如，如果您在其他 AWS 账户中选择了目标存储桶，而没有权限读取和写入存储桶策略，则您会看到以下消息。您必须让目标存储桶的拥有者将显示的存储桶策略添加到目标存储桶中。如果策略未添加到目标存储桶中，则您不会获得导出数据，因为 Amazon S3 无权写入目标存储桶。如果源存储桶属于其他账户而非当前用户，则在策略中必须替换掉源存储桶的正确账户 ID。

有关导出的数据以及筛选器工作原理的信息，请参阅 Amazon S3 分析 – 存储类分析。

使用 REST API

要使用 REST API 配置存储类分析，请使用 PutBucketAnalyticsConfiguration。您还可以将等效操作与 AWS CLI 或 AWS SDK 结合使用。

您可以使用以下 REST API 来处理存储类分析：

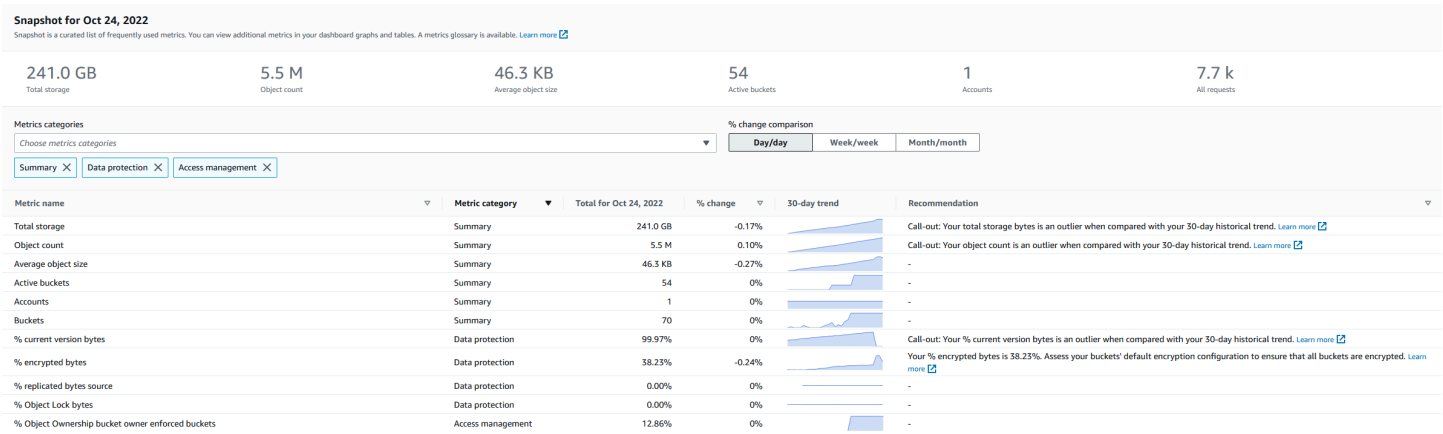
- DELETE Bucket Analytics configuration
- GET Bucket Analytics configuration
- List Bucket Analytics Configuration

使用 Amazon S3 Storage Lens 存储统计管理工具评估您的存储活动和使用情况

Amazon S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储和活动。S3 Storage Lens 存储统计管理工具还分析指标以提供上下文建议，您可以使用这些建议来优化存储成本并应用最佳实践来保护数据。

您可以使用 S3 Storage Lens 存储统计管理工具指标生成汇总见解。例如，您可以了解整个组织中有多少存储空间，或增长最快的桶和前缀是哪些。您还可以使用 S3 Storage Lens 存储统计管理工具指标来识别成本优化机会，实施数据保护和访问管理最佳实践，并提高应用程序工作负载的性能。例如，您可以识别没有 S3 生命周期规则的存储桶，以中止超过 7 天的未完成分段上传。您还可以识别未遵循数据保护最佳实践（例如使用 S3 复制或 S3 版本控制）的桶。

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、存储桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 存储桶。



S3 Storage Lens 存储统计管理工具指标和功能

S3 Storage Lens 存储统计管理工具提供每日更新的交互式原定设置控制面板。S3 Storage Lens 存储统计管理工具预配置此控制面板，以直观显示整个账户的见解和趋势，并在 S3 控制台中每天更新这些信息。此控制面板中的指标也会在 Buckets（桶）页面上的账户快照进行汇总。有关更多信息，请参阅默认控制面板。

要创建其他控制面板并按 AWS 区域、S3 桶或账户（对于 AWS Organizations）限定其范围，您可以创建 S3 Storage Lens 存储统计管理工具控制面板配置。您可以使用 Amazon S3 控制台、AWS Command Line Interface (AWS CLI)、AWS SDK 或 Amazon S3 REST API 创建和管理 S3 Storage Lens 存储统计管理工具控制面板配置。创建或编辑 S3 Storage Lens 存储统计管理工具控制面板时，定义控制面板范围和指标选择。

S3 Storage Lens 存储统计管理工具提供免费指标、高级指标和建议，您可以支付额外费用升级到高级指标和建议。借助高级指标和建议，您可以访问其他指标和功能，以深入了解您的存储。这些功能包括高级指标类别、前缀聚合、上下文建议和 Amazon CloudWatch 发布。前缀聚合和上下文建议仅在 Amazon S3 控制台中提供。有关 S3 Storage Lens 存储统计管理工具定价的信息，请参阅 Amazon S3 定价。

指标类别

在免费和高级套餐中，指标划分为与关键使用案例一致的类别，例如成本优化和数据保护。免费指标包括摘要、成本优化、数据保护、访问管理、性能和事件指标。升级到高级指标和建议时，可以启用高级成本优化和数据保护指标。您可以使用这些高级指标进一步降低您的 S3 存储成本并改善您的数据保护态势。您还可以启用活动指标和详细的状态代码指标，以提高访问您的 S3 桶的应用程序工作负载的性能。有关免费和高级指标类别的更多信息，请参阅指标选择。

您可以根据 S3 最佳实践来评估存储，例如分析启用了加密或 S3 对象锁定或 S3 版本控制的桶的百分比。您还可以确定潜在的成本节约机会。例如，您可以使用 S3 生命周期规则计数指标来识别缺少生命

周期过期或转换规则的桶。此外，您还可以分析每个桶的请求活动，以查找可以将对象转换为较低成本存储类的桶。有关更多信息，请参阅 Amazon S3 Storage Lens 存储统计管理工具使用案例。

指标导出

除了查看 S3 控制台中的控制面板之外，您还可以将指标以 CSV 或 Parquet 格式导出至 S3 桶中，以使用您选择的分析工具进行进一步分析。有关更多信息，请参阅 使用数据导出查看 Amazon S3 Storage Lens 存储统计管理工具指标。

Amazon CloudWatch 发布

您可以将 S3 Storage Lens 存储统计管理工具使用情况和活动指标发布到 Amazon CloudWatch，以便在 CloudWatch dashboards（控制面板）中创建运营状况的统一视图。您还可以使用 CloudWatch 功能（如告警和触发操作、指标数学和异常检测）来监控 S3 Storage Lens 存储统计管理工具指标并采取措施。此外，CloudWatch API 操作使应用程序（包括第三方提供商）能够访问 S3 Storage Lens 存储统计管理工具指标。CloudWatch 发布选项可用于升级到 S3 Storage Lens 存储统计管理工具高级指标和建议。有关在 CloudWatch 中 S3 Storage Lens 存储统计管理工具支持的更多信息，请参阅 在 CloudWatch 中监控 S3 Storage Lens 存储统计管理工具指标。

有关使用 S3 Storage Lens 存储统计管理工具的更多信息，请参阅以下主题。

主题

了解 Amazon S3 Storage Lens 存储统计管理工具

> **Important**
>
> Amazon S3 现在将具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）作为 Amazon S3 中每个存储桶的基本加密级别。从 2023 年 1 月 5 日起，上传到 Amazon S3 的所有新对象

都将自动加密，不会产生额外费用，也不会影响性能。S3 存储桶默认加密配置和上传的新对象的自动加密状态可在 AWS CloudTrail 日志、S3 清单、S3 Storage Lens 存储统计管理工具、Amazon S3 控制台中获得，并可用作 AWS Command Line Interface 和 AWS SDK 中的附加 Amazon S3 API 响应标头。有关更多信息，请参阅默认加密常见问题解答。

Amazon S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。您可以使用 S3 Storage Lens 存储统计管理工具指标生成摘要见解，例如，了解整个组织中有多少存储空间，或增长最快的桶和前缀是哪些。您还可以使用 S3 Storage Lens 存储统计管理工具指标来识别成本优化机会，实施数据保护和安全最佳实践，并提高应用程序工作负载的性能。例如，您可以识别没有 S3 生命周期规则的桶，使超过 7 天的未完成分段上传过期。您还可以识别未遵循数据保护最佳实践（例如使用 S3 复制或 S3 版本控制）的桶。S3 Storage Lens 存储统计管理工具还分析指标以提供上下文建议，您可以使用这些建议来优化存储成本并应用最佳实践来保护数据。

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、存储桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 存储桶。您可以使用 Amazon S3 控制台、AWS Command Line Interface (AWS CLI)、AWS SDK 或 Amazon S3 REST API 创建和管理 S3 Storage Lens 存储统计管理工具控制面板。

S3 Storage Lens 存储统计管理工具的概念和术语

本部分包含成功了解和使用 Amazon S3 Storage Lens 存储统计管理工具所必需的术语和概念。

主题

- 建议
- 指标选择
- S3 Storage Lens 存储统计管理工具和 AWS Organizations

控制面板配置

S3 Storage Lens 存储统计管理工具需要控制面板配置，其中包含代表您聚合指标所需的属性，以供单个控制面板或导出使用。创建配置时，您可以选择控制面板名称和主区域，创建控制面板后无法对其进行更改。您可以选择添加标签并配置以 CSV 或 Parquet 格式导出指标。

在控制面板配置中，您还可以定义控制面板范围和指标选择。范围可以包括您的组织账户或按区域、桶和账户筛选的分区的所有存储。配置指标选择时，您可以在免费指标与高级指标和建议之间进行选择，您可以额外付费升级到高级指标和建议。借助高级指标和建议，您可以访问其他指标和功能。这些功能包括高级指标类别、前缀级聚合、上下文建议和 Amazon CloudWatch 发布。有关 S3 Storage Lens 存储统计管理工具定价的信息，请参阅 Amazon S3 定价。

默认控制面板

控制台上的 S3 Storage Lens 存储统计管理工具默认控制面板命名为 default-account-dashboard。S3 预配置此控制面板，以直观显示整个账户的见解和趋势，并在 S3 控制台中每天更新这些信息。您无法修改原定设置控制面板的配置范围，但可以将指标选择从免费指标升级到高级指标和建议。您可以配置可选指标导出，甚至可以禁用控制面板。但是，您无法删除默认控制面板。

> **Note**
>
> 如果您禁用原定设置控制面板，则它将不再更新。您将不会再在 S3 Storage Lens 存储统计管理工具控制面板、指标导出或 S3 桶页面上的账户快照中收到任何新的每日指标。如果控制面板使用高级指标和建议，则不再向您收费。在数据查询的 14 天期限到期之前，您仍然可以在控制面板中查看历史数据。如果您启用了高级指标和建议，则此期限为 15 个月。要访问历史数据，您可以在到期期限内重新启用控制面板。

控制面板

您可以创建其他 S3 Storage Lens 存储统计管理工具控制面板，并按 AWS 区域、S3 桶或账户（对于 AWS Organizations）限定其范围。创建或编辑 S3 Storage Lens 存储统计管理工具控制面板时，定义控制面板范围和指标选择。S3 Storage Lens 存储统计管理工具提供免费指标、高级指标和建议，您可以支付额外费用升级到高级指标和建议。借助高级指标和建议，您可以访问其他指标和功能，以深入了

解您的存储。这些功能包括高级指标类别、前缀级聚合、上下文建议和 Amazon CloudWatch 发布。有关 S3 Storage Lens 存储统计管理工具定价的信息，请参阅 Amazon S3 定价。

您还可以禁用或删除控制面板。如果禁用控制面板，它将不再更新，并且您将不再收到任何新的每日指标。在 14 天到期之前，您仍然可以查看历史数据。如果您为该控制面板启用了高级指标和建议，则此期限为 15 个月。要访问历史数据，您可以在到期期限内重新启用控制面板。

如果删除控制面板，将丢失所有控制面板的配置设置。您将不再收到任何新的每日指标，而且您也无法访问与该控制面板关联的历史数据。如果要访问已删除控制面板的历史数据，则必须在同一主区域中创建另一个具有相同名称的控制面板。

> **Note**
>
> - 您可以使用 S3 Storage Lens 存储统计管理工具为每个主区域创建多达 50 个控制面板。
> - 组织级控制面板只能限于区域范围。

账户快照

S3 Storage Lens 存储统计管理工具的 Account snapshot（账户快照）汇总了原定设置控制面板中的指标，并在 S3 控制台的 Buckets（桶）页面上显示总存储空间、对象计数和平均对象大小。账户快照让您可以快速获取有关存储的见解，而不必离开 Buckets（桶）页面。账户快照还允许您一键访问您的交互式 S3 Storage Lens 存储统计管理工具控制面板。

您可以使用控制面板可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、桶、对象或前缀级别生成见解。您还可以按 CSV 或 Parquet 格式将每天一次的指标导出发送到 S3 桶。

因为 default-account dashboard（原定设置账户控制面板）已链接到 Account snapshot（账户快照），所以您无法修改其控制面板范围。但是，在 default-account-dashboard（原定设置账户控制面板）中，您可以将指标选择从免费指标升级到付费的高级指标和建议。升级后，您可以在 S3 Storage Lens 存储统计管理工具 Account snapshot（账户快照）中显示所有请求、上载的字节数和下载的字节数。

> **Note**
>
> 如果禁用原定设置控制面板，将不再更新您的 Account snapshot（账户快照）。要在 Account snapshot（账户快照）中继续显示指标，您可以重新启用 default-account-dashboard（原定设置账户控制面板）。

指标导出

S3 Storage Lens 存储统计管理工具指标导出是一个包含 S3 Storage Lens 存储统计管理工具配置中标识的所有指标的文件。此信息每天以 CSV 或 Parquet 格式生成，并发送到 S3 桶。您可以通过所选的指标工具使用指标导出进行进一步分析。用于指标导出的 S3 桶必须与 S3 Storage Lens 存储统计管理工具配置位于同一区域。您可以通过编辑控制面板配置，从 S3 控制台生成 S3 Storage Lens 存储统计管理工具指标导出。您也可以使用 AWS CLI 和 AWS SDK 配置指标导出。

主区域

主区域是在其中存储给定控制面板配置的所有 Amazon S3 Storage Lens 存储统计管理工具指标的 AWS 区域。创建 S3 Storage Lens 存储统计管理工具控制面板配置时，必须选择主区域。选择主区域后，无法对其进行更改。此外，如果您要创建 Storage Lens 组，我们建议您选择与 Storage Lens 存储统计管理工具控制面板相同的主区域。

> **Note**
>
> 您可以选择以下一个区域作为主区域：
>
> - 美国东部（弗吉尼亚州北部）– us-east-1
> - 美国东部（俄亥俄州）– us-east-2
> - 美国西部（加利福尼亚北部）– us-west-1
> - 美国西部（俄勒冈州）– us-west-2
> - 亚太地区（孟买）–(ap-south-1)
> - 亚太地区（首尔）–(ap-northeast-2)
> - 亚太地区（新加坡）–(ap-southeast-1)
> - 亚太地区（悉尼）– ap-southeast-2
> - 亚太地区（东京）–(ap-northeast-1)
> - 加拿大（中部）– ca-central-1
> - 中国（北京）– cn-north-1
> - 中国（宁夏）– cn-northwest-1
> - 欧洲地区（法兰克福）– eu-central-1
> - 欧洲地区（爱尔兰）– eu-west-1
> - 欧洲地区（伦敦）– eu-west-2
> - 欧洲地区（巴黎）– eu-west-3

- 欧洲地区（斯德哥尔摩）– eu-north-1
- 南美洲（圣保罗）– (sa-east-1)

保留期

S3 Storage Lens 存储统计管理工具指标将保留，以便您可以查看历史趋势并比较存储和活动随时间推移的差异。您可以使用 Amazon S3 Storage Lens 存储统计管理工具指标以进行查询，因此您可以查看历史趋势并比较存储使用情况和活动随时间推移的差异。

所有 S3 Storage Lens 存储统计管理工具指标将保留 15 个月。但是，指标仅适用于特定持续时间的查询，这取决于您的指标选择。无法修改此持续时间。免费指标可用于 14 天的查询，高级指标可用于 15 个月的查询。

指标类别

在免费和高级套餐中，S3 Storage Lens 存储统计管理工具指标划分为与关键使用案例一致的类别，例如成本优化和数据保护。免费指标包括摘要、成本优化、数据保护、访问管理、性能和事件指标。升级到高级指标和建议时，您可以启用额外的成本优化和数据保护指标，这些指标可用于进一步降低 S3 存储成本和确保数据受到保护。您还可以启用活动指标和详细的状态代码指标，这些指标可用于提高应用程序工作流的性能。

以下列表显示了所有免费和高级指标类别。有关每个类别中包含的各个指标的完整列表，请参阅指标词汇表。

汇总指标

摘要指标提供有关您的 S3 存储的一般见解，包括您的总存储字节数和对象计数。

成本优化指标

成本优化指标提供见解，您可以使用这些见解来管理和优化存储成本。例如，您可以识别具有超过 7 天的未完成分段上传的桶。

借助高级指标和建议，您可以启用高级成本优化指标。这些指标包括 S3 生命周期规则计数指标，您可以使用这些指标来获取每个桶的到期时间和转换 S3 生命周期规则计数。

数据保护指标

数据保护指标提供对于加密和 S3 版本控制等数据保护功能的见解。您可以使用这些指标来识别未遵循数据保护最佳实践的桶。例如，您可以识别未使用具有 AWS Key Management Service 密钥的原定设置加密（SSE-KMS）或 S3 版本控制的桶。

借助高级指标和建议，您可以启用高级数据保护指标。这些指标包括每个桶的复制规则计数指标。

访问管理指标

访问管理指标提供 S3 对象所有权的见解。您可以使用这些指标来查看您的桶使用了哪些对象所有权设置。

事件指标

事件指标提供 S3 事件通知的见解。通过事件指标，您可以查看哪些桶配置了 S3 事件通知。

性能指标

性能指标提供 S3 Transfer Acceleration 的见解。通过性能指标，您可以查看哪些桶启用了 Transfer Acceleration。

活动指标（高级）

如果您将控制面板升级到高级指标和建议，则可以启用活动指标。活动指标提供有关如何请求存储（例如，All 请求、Get 请求、Put 请求）、上传或下载的字节数以及错误的详细信息。

前缀级别的活动指标可用于帮助您确定哪些前缀不经常使用，以便您可以使用 S3 生命周期过渡到更适合的存储类。

详细状态代码指标（高级）

如果您将控制面板升级到高级指标和建议，则可以启用详细状态代码指标。详细状态代码指标提供有关 HTTP 状态代码的见解，例如“403 禁止”和“503 服务不可用”，您可以使用这些信息来排查访问或性能问题。例如，您可以查看 403 Forbidden error count（403 禁止错误计数）指标，以确定在未应用正确权限的情况下访问桶的工作负载。

前缀级别的详细状态代码指标可用于更好地了解按前缀划分的 HTTP 状态代码出现次数。例如，“503 错误计数”指标使您能够识别在数据摄取期间收到节流请求的前缀。

建议

S3 Storage Lens 存储统计管理工具提供自动建议来帮助您优化存储。在 S3 Storage Lens 存储统计管理工具控制面板中，建议与相关指标一起放置在上下文中。历史数据不符合建议的条件，因为建议与最近一段时期的情况相关。仅在建议相关时才会显示它们。

S3 Storage Lens 存储统计管理工具建议有以下形式：

- 建议

建议会提醒您注意存储和活动中的趋势，这些趋势可能表明存储成本优化机会或可采用某种数据保护最佳实践。您可以使用《Amazon S3 用户指南》和 S3 Storage Lens 存储统计管理工具控制面板中的建议主题，深入了解有关特定区域、桶或前缀的更多详细信息。

- 标注

  标注是一些建议，可提醒您注意在一段时间内存储和活动中存在的值得留意的异常，您可能需要进一步关注或监控这些异常。

  - 异常标注

    S3 Storage Lens 存储统计管理工具根据您最近 30 天的趋势，为异常的指标提供了标注。异常是使用标准分数（也称为 z 得分）计算的。在这个分数中，将从该指标过去 30 天的平均值中减去当天的指标。然后用当天的指标除以该指标在过去 30 天内的标准差。得出的分数通常在 -3 到 +3 之间。此数字代表当天的指标与平均值相比的标准差。

    S3 Storage Lens 存储统计管理工具将分数 > 2 或 <-2 的指标视为异常，因为它们高于或低于正态分布数据的 95%。

  - 重大更改标注

    重大更改标注适用于预计更改频率较低的指标。因此，重大更改标注的灵敏度设置为高于异常值计算（异常值计算的浮动相对于前一天、前一周或前个月通常介于 +/-20% 的范围内）。

    解决存储和活动中的标注问题 – 如果您收到重大更改标注，这不一定意味着出现问题。该标注可能是由于您的存储中发生预期变化所致。例如，您最近可能添加了大量新对象、删除了大量对象或进行了类似的计划更改。

    如果您在控制面板上看到重大更改标注，请记下它并确定是否可以根据最近的情况来解释。如果没有，请使用 S3 Storage Lens 存储统计管理工具控制面板深入了解更多详细信息，以了解导致波动的特定区域、桶或前缀。

- 提醒

  提醒可以深入了解 Amazon S3 的工作原理。它们可以帮助您详细了解如何使用 S3 功能来降低存储成本或应用数据保护最佳实践。

指标选择

S3 Storage Lens 存储统计管理工具提供了两个指标选择，您可以为控制面板和导出进行选择：免费指标和高级指标和建议。

- 免费指标

  S3 Storage Lens 存储统计管理工具为所有控制面板和配置提供免费指标。免费指标包含与存储相关的指标，例如桶数量和账户中的对象数。免费指标还包括基于使用案例的指标（例如，成本优化和数据保护指标），您可以使用这些指标来调查您的存储配置是否符合 S3 最佳实践。所有免费指标每天收集。数据可用于查询达 14 天。有关免费指标可提供哪些指标的更多信息，请参阅Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

- 高级指标和建议

  S3 Storage Lens 存储统计管理工具为所有控制面板和配置提供免费指标以及升级到高级指标和建议选项的选择。将收取额外的 费用。有关更多信息，请参阅 Amazon S3 定价。

  高级指标和建议包括免费指标中的所有指标以及其他指标，例如高级数据保护和成本优化指标、活动指标和详细的状态代码指标。高级指标和建议还提供了帮助您优化存储的建议。建议与控制面板中的相关指标一起放置在上下文中。

  高级指标和建议包括以下功能：

  - 高级指标 - 生成其他指标。有关高级指标类别的完整列表，请参阅指标类别。要获得指标的完整列表，请参阅Amazon S3 Storage Lens 存储统计管理工具指标词汇表。
  - Amazon CloudWatch 发布 – 将 S3 Storage Lens 存储统计管理工具指标发布到 CloudWatch，以便在 CloudWatch 控制面板中创建运营状况的统一视图。您还可以使用 CloudWatch 操作和功能（如告警和触发操作、指标数学和异常检测）来监控 S3 Storage Lens 存储统计管理工具指标并采取措施。有关更多信息，请参阅 在 CloudWatch 中监控 S3 Storage Lens 存储统计管理工具指标。
  - 前缀聚合 – 收集前缀级别的指标。启用前缀聚合可在前缀级别扩展控制面板配置中包含的所有指标。仅为满足所配置阈值的前缀生成指标。请注意，适用于前缀级别的指标可用于前缀聚合，但桶级别的设置和规则计数指标除外。前缀级别的指标不会发布到 CloudWatch。
  - Storage Lens 组聚合 – 收集 Storage Lens 组级别的指标。启用高级指标和建议以及 Storage Lens 组聚合后，您可以指定要在 Storage Lens 控制面板中包括或排除哪些 Storage Lens 组。必须至少指定一个 Storage Lens 组。指定的 Storage Lens 组还必须位于控制面板账户中指定的主区域内。Storage Lens 组级别的指标不会发布到 CloudWatch。

  所有高级指标每天收集。数据最长在 15 个月内可供查询。有关 S3 Storage Lens 存储统计管理工具聚合的存储指标的更多信息，请参阅Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

> **Note**
>
> 仅当在 Amazon S3 控制台上使用 S3 Storage Lens 存储统计管理工具控制面板时建议才可用。

S3 Storage Lens 存储统计管理工具和 AWS Organizations

AWS Organizations 是一项 AWS 服务，可帮助您将所有 AWS 账户聚合在一个组织层次结构下。Amazon S3 Storage Lens 存储统计管理工具与 AWS Organizations 配合使用，可提供整个 Amazon S3 存储范围内对象存储和活动的单一视图。

有关更多信息，请参阅 将 Amazon S3 Storage Lens 存储统计管理工具与 AWS Organizations 结合使用。

- 可信访问权限

  使用组织的管理账户，您必须为 S3 Storage Lens 存储统计管理工具启用可信访问权限，以聚合组织中所有成员账户的存储指标和使用情况数据。然后，您可以使用管理账户或向组织中其他账户授予委托管理员访问权限，为组织创建控制面板或导出。

  您可以随时为 S3 Storage Lens 存储统计管理工具禁用可信访问权限，这将阻止 S3 Storage Lens 存储统计管理工具聚合组织的指标。

- 委托管理员

  您可以使用 AWS Organizations 管理账户或向组织中其他账户授予委托管理员访问权限，为您的组织创建 S3 Storage Lens 存储统计管理工具的控制面板和指标。您可以随时取消注册委托管理员。取消注册委托管理员还会自动阻止该委托管理员创建的所有组织级控制面板聚合新的存储指标。

有关更多信息，请参阅《AWS Organizations 用户指南》中的 Amazon S3 Storage Lens 存储统计管理工具和 AWS Organizations。

Amazon S3 Storage Lens 存储统计管理工具服务相关角色

除了 AWS Organizations 可信访问权限外，Amazon S3 Storage Lens 存储统计管理工具还使用 AWS Identity and Access Management（IAM）服务相关角色。服务相关角色是一种与 S3 Storage Lens 存储统计管理工具直接关联的独特类型的 IAM 角色。S3 Storage Lens 存储统计管理工具预定义了服务相关角色，并包括从组织中的成员账户收集每日存储和活动指标所需的所有权限。

有关更多信息，请参阅使用面向 Amazon S3 Storage Lens 存储统计管理工具的服务相关角色。

将 Amazon S3 Storage Lens 存储统计管理工具与 AWS Organizations 结合使用

Amazon S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。您可以使用 S3 Storage Lens 存储统计管理工具指标生成摘要见解，例如，了解整个组织中有多少存储空间，或增长最快的桶和前缀是哪些。您还可以使用 S3 Storage Lens 存储统计管理工具指标来识别成本优化机会，实施数据保护和安全最佳实践，并提高应用程序工作负载的性能。例如，您可以识别没有 S3 生命周期规则的桶，使超过 7 天的未完成分段上传过期。您还可以识别未遵循数据保护最佳实践（例如使用 S3 复制或 S3 版本控制）的桶。S3 Storage Lens 存储统计管理工具还分析指标以提供上下文建议，您可以使用这些建议来优化存储成本并应用最佳实践来保护数据。

您可以使用 Amazon S3 Storage Lens 存储统计管理工具收集属于 AWS Organizations 层次结构的所有 AWS 账户 的存储指标和使用情况数据。为此，您必须使用 AWS Organizations，并且必须使用您的 AWS Organizations 管理账户启用 S3 Storage Lens 存储统计管理工具可信访问权限。

启用可信访问权限后，您可以添加对组织中账户的委托管理员访问权限。然后，这些账户可以创建 S3 Storage Lens 存储统计管理工具配置和控制面板，以收集组织范围的存储指标和用户数据。

有关启用可信访问权限的更多信息，请参阅《AWS Organizations 用户指南》中的 Amazon S3 Storage Lens 存储统计管理工具和 AWS Organizations。

主题

为 S3 Storage Lens 存储统计管理工具启用可信访问权限

通过启用可信访问权限，即可允许 Amazon S3 Storage Lens 存储统计管理工具通过 AWS Organizations API 操作访问您的 AWS Organizations 层次结构、成员资格和结构。然后，S3 Storage Lens 存储统计管理工具便成为整个组织结构的可信服务。

无论何时创建控制面板配置，S3 Storage Lens 存储统计管理工具都会在组织的管理或委托管理员账户中创建服务相关角色。服务相关角色授予 S3 Storage Lens 存储统计管理工具执行以下操作的权限：

- 描述组织
- 列出账户
- 验证组织的 AWS 服务访问权限列表
- 为组织获取委托管理员

随后，S3 Storage Lens 存储统计管理工具可以确保它具有访问权限，以收集组织中账户的跨账户指标。有关更多信息，请参阅使用面向 Amazon S3 Storage Lens 存储统计管理工具的服务相关角色。

启用可信访问权限后，您可以为组织中的账户分配委托管理员访问权限。当账户被标记为服务的委托管理员时，该账户将获得访问所有只读组织 API 操作的授权。此访问权限向委托管理员提供了对您组织成员和结构的可见性，以便他们也可以创建 S3 Storage Lens 存储统计管理工具控制面板。

> **Note**
>
> 只有管理账户才能为 Amazon S3 Storage Lens 存储统计管理工具启用可信访问权限。

为 S3 Storage Lens 存储统计管理工具禁用可信访问权限

通过禁用可信访问权限，您将 S3 Storage Lens 存储统计管理工具限制为仅在账户级别工作。此外，每个账户持有人所看到的 S3 Storage Lens 存储统计管理工具信息仅限于其账户范围，而不是整个组织。任何需要可信访问权限的控制面板将不再更新，但它们将能够在数据可供查询的相应时间段内保留它们的历史数据。

> **Note**
>
> - 禁用对 S3 Storage Lens 存储统计管理工具的可信访问还会自动阻止所有组织级控制面板收集和聚合存储指标。
> - 在数据可供查询期间，您的管理和委托管理员账户仍然能够看到您现有的组织级控制面板的历史数据。

为 S3 Storage Lens 存储统计管理工具注册委托管理员

您可以使用组织的管理账户或委托管理员账户创建组织级控制面板。委托管理员账户允许除管理账户以外的其他账户创建组织级控制面板。只有组织的管理账户才能将其他账户注册为组织的委托管理员以及取消其注册。

要使用 Amazon S3 控制台注册委托管理员，请参阅为 S3 Storage Lens 存储统计管理工具注册委托管理员。

您还可以使用管理账户中的 AWS Organizations REST API、AWS CLI 或 SDK 注册委托管理员。有关更多信息，请参阅《AWS Organizations API 参考》中的 RegisterDelegatedAdministrator。

> **Note**
>
> 在使用 AWS Organizations REST API、AWS CLI 或 SDK 指定委托管理员之前，必须调用 EnableAWSOrganizationsAccess 操作。

为 S3 Storage Lens 存储统计管理工具取消注册委托管理员

您还可以取消注册委托管理员账户。委托管理员账户允许除管理账户以外的其他账户创建组织级控制面板。只有组织的管理账户才能以组织的委托管理员身份取消注册账户。

要使用 S3 控制台取消注册委托管理员，请参阅为 S3 Storage Lens 存储统计管理工具取消注册委托管理员。

您还可以使用管理账户中的 AWS Organizations REST API、AWS CLI 或 SDK 取消注册委托管理员。有关更多信息，请参阅《AWS Organizations API 参考》中的 DeregisterDelegatedAdministrator。

> **Note**
>
> - 取消注册委托管理员还会自动阻止该委托管理员创建的所有组织级控制面板聚合新的存储指标。
> - 当数据可用于查询时，取消注册的委派管理员仍然可以查看他们创建的控制面板的历史数据。

Amazon S3 Storage Lens 存储统计管理工具权限

Amazon S3 Storage Lens 存储统计管理工具需要 AWS Identity and Access Management（IAM）中的新权限才能授权访问 S3 Storage Lens 存储统计管理工具操作。要授予这些权限，您可以使用基于身份的 IAM policy。您可以将此策略附加到 IAM 用户、组或角色，以授予其权限。此类权限可能包括启用或禁用 S3 Storage Lens 存储统计管理工具或访问任何 S3 Storage Lens 存储统计管理工具控制面板或配置。

除非同时满足以下两个条件，否则 IAM 用户或角色必须属于创建或拥有控制面板或配置的账户：

- 您的账户是 AWS Organizations 的成员。
- 您获得了相关访问权限，可以使用您的管理账户作为委托管理员，创建组织级别的控制面板。

> **Note**
>
> - 您不能使用账户的根用户凭证查看 Amazon S3 Storage Lens 存储统计管理工具控制面板。要访问 S3 Storage Lens 存储统计管理工具控制面板，您必须向新的或现有的 IAM 用户授予所需的 IAM 权限。然后，使用这些用户凭证登录以访问 S3 Storage Lens 存储统计管理工具控制面板。有关更多信息，请参阅《IAM 用户指南》中的 IAM 安全最佳实践。
> - 在 Amazon S3 控制台上使用 S3 Storage Lens 存储统计管理工具可能需要多个权限。例如，要在控制台上编辑控制面板，您需要以下权限：
>   - `s3:ListStorageLensConfigurations`
>   - `s3:GetStorageLensConfiguration`
>   - `s3:PutStorageLensConfiguration`

主题

设置账户权限以使用 S3 Storage Lens 存储统计管理工具

要创建和管理 S3 Storage Lens 存储统计管理工具控制面板和 Storage Lens 存储统计管理工具控制面板配置，您必须拥有以下权限，具体取决于您要执行的操作：

Amazon S3 Storage Lens 存储统计管理工具相关的 IAM 权限

操作	IAM 权限
在 Amazon S3 控制台中创建或更新 S3 Storage Lens 存储统计管理工具控制面板。	s3:ListStorageLensConfigurations s3:GetStorageLensConfiguration s3:GetStorageLensConfigurationTagging s3:PutStorageLensConfiguration s3:PutStorageLensConfigurationTagging
在 Amazon S3 控制台上获取 S3 Storage Lens 存储统计管理工具控制面板的标签。	s3:ListStorageLensConfigurations s3:GetStorageLensConfigurationTagging
在 Amazon S3 控制台上查看 S3 Storage Lens 存储统计管理工具控制面板。	s3:ListStorageLensConfigurations s3:GetStorageLensConfiguration s3:GetStorageLensDashboard
在 Amazon S3 控制台上删除 S3 Storage Lens 存储统计管理工具控制面板。	s3:ListStorageLensConfigurations s3:GetStorageLensConfiguration s3:DeleteStorageLensConfiguration
在 AWS CLI 或 AWS SDK 中创建或更新 S3 Storage Lens 存储统计管理工具配置。	s3:PutStorageLensConfiguration s3:PutStorageLensConfigurationTagging
使用 AWS CLI 或 AWS SDK 获取 S3 Storage Lens 存储统计管理工具配置的标签。	s3:GetStorageLensConfigurationTagging

操作	IAM 权限
使用 AWS CLI 或 AWS SDK 查看 S3 Storage Lens 存储统计管理工具配置。	`s3:GetStorageLensConfiguration`
使用 AWS CLI 或 AWS SDK 删除 S3 Storage Lens 存储统计管理工具配置。	`s3:DeleteStorageLensConfiguration`

> **Note**
>
> - S3 Storage Lens 存储分析功能控制面板视图使用事件名称 `GetStorageLensDashboardDataInternal` 记录在 CloudTrail 中。
> - 您可以使用 IAM policy 中的资源标签来管理权限。
> - 具有这些权限的 IAM 用户或角色可以查看他们可能不具备从中读取或列出对象的直接权限的桶和前缀中的指标。
> - 对于启用了前缀级别指标的 S3 Storage Lens 存储统计管理工具控制面板，如果选定的前缀路径与某个对象键相匹配，则控制面板可能会将该对象键显示为另一个前缀。
> - 对于存储在账户的桶中的指标导出，使用 IAM policy 中的现有 `s3:GetObject` 权限进行权限授权。同样，对于 AWS Organizations 实体，组织的管理账户或委托管理员账户可以使用 IAM policy 来管理组织级控制面板和配置的访问权限。

设置账户权限以使用 S3 Storage Lens 组

您可以使用 S3 Storage Lens 组，根据前缀、后缀、对象标签、对象大小或对象期龄来了解桶内的存储分布。您可以将 Storage Lens 组附加到控制面板以查看其聚合指标。

要使用 Storage Lens 组，你需要具备某些权限。有关更多信息，请参阅 the section called “Storage Lens 组权限”。

设置将 S3 Storage Lens 存储统计管理工具与 AWS Organizations 结合使用的权限

您可以使用 Amazon S3 Storage Lens 存储统计管理工具收集属于 AWS Organizations 层次结构的所有账户的存储指标和使用情况数据。以下是与在组织中使用 S3 Storage Lens 存储统计管理工具相关的操作和权限。

使用 S3 Storage Lens 存储统计管理工具的 AWS Organizations 相关 IAM 权限

操作	IAM 权限
为您的组织启用 S3 Storage Lens 存储统计管理工具的可信访问权限。	`organizations:EnableAWSServiceAccess`
为您的组织禁用 S3 Storage Lens 存储统计管理工具的可信访问权限。	`organizations:DisableAWSServiceAccess`
注册委托管理员，为您的组织创建 S3 Storage Lens 存储统计管理工具控制面板或配置。	`organizations:RegisterDelegatedAdministrator`
取消注册委托管理员，这样他们就无法再为您的组织创建 S3 Storage Lens 存储统计管理工具控制面板或配置。	`organizations:DeregisterDelegatedAdministrator`
创建 S3 Storage Lens 存储统计管理工具组织范围配置的其他权限。	`organizations:DescribeOrganization` `organizations:ListAccounts` `organizations:ListAWSServiceAccessForOrganization` `organizations:ListDelegatedAdministrators` `iam:CreateServiceLinkedRole`

使用 Amazon S3 Storage Lens 存储统计管理工具查看指标

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 桶。

原定设置情况下，所有控制面板都配置了免费指标，其中包括可用于了解 S3 存储中的使用情况和活动、优化存储成本以及实施数据保护和访问管理最佳实践的指标。免费指标向下聚合到桶级别。使用免费指标，数据最多在 14 天内可供查询。

高级指标和建议包括以下附加功能，您可以使用这些功能进一步了解跨存储的使用情况和活动以及优化存储的最佳实践：

- 上下文建议（仅在控制台中提供）
- 高级指标（包括按桶聚合的活动指标）
- Prefix aggregation（前缀聚合）
- Storage Lens 组聚合
- Storage Lens 组聚合
- Amazon CloudWatch 发布

高级指标数据可供查询 15 个月。使用包含高级指标的 S3 Storage Lens 存储统计管理工具需要额外收费。有关更多信息，请参阅 Amazon S3 定价。有关免费和高级指标的更多信息，请参阅指标选择。

主题

- 在控制面板上查看 S3 Storage Lens 存储统计管理工具指标
- 使用数据导出查看 Amazon S3 Storage Lens 存储统计管理工具指标
- 在 CloudWatch 中监控 S3 Storage Lens 存储统计管理工具指标

在控制面板上查看 S3 Storage Lens 存储统计管理工具指标

在 Amazon S3 控制台中，S3 Storage Lens 存储统计管理工具提供了一个交互式默认控制面板，可用于可视化数据中的见解和趋势。您还可以使用此控制面板标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在账户、桶、AWS 区域、前缀或 Storage Lens 组级别生成见解。如果您启用了 S3 Storage Lens 存储统计管理工具以与 AWS Organizations 配合使用，您还可以生成组织级别的见解（例如 AWS Organizations 层次结构中所有账户的数据）。控制面板始终加载指标可用的最新日期。

控制台上的 S3 Storage Lens 存储统计管理工具默认控制面板命名为 default-account-dashboard。Amazon S3 会预配置此控制面板，以可视化整个账户的汇总见解和趋势，并在 S3 控制台中每天更新这些信息。您无法修改默认控制面板的配置范围，但可以将指标选择从免费指标升级到付费的高级指标和建议。借助高级指标和建议，您可以访问其他指标和功能。这些功能包括高级指标类别、前缀级聚合、上下文建议和 Amazon CloudWatch 发布。

您可以禁用原定设置控制面板，但不能删除它。如果您禁用原定设置控制面板，则它将不再更新。您也将不会再在 S3 Storage Lens 存储统计管理工具或存续桶页面上的账户快照部分收到任何新的每日指标。在数据查询的 14 天期限到期之前，您仍然可以在原定设置控制面板中查看历史数据。如果您启用了高级指标和建议，则此期限为 15 个月。要访问此数据，您可以在有效期内重新启用原定设置控制面板。

您可以创建其它 S3 Storage Lens 存储统计管理工具控制面板，并按 AWS 区域、S3 桶或账户限定其范围。如果您启用了 Storage Lens 存储统计管理工具以与 AWS Organizations 配合使用，则还可以按组织限定控制面板的范围。创建或编辑 S3 Storage Lens 存储统计管理工具控制面板时，定义控制面板范围和指标选择。

您可以禁用或删除您创建的任何其他控制面板。

- 如果禁用控制面板，它将不再更新，并且您将不再收到任何新的每日指标。在 14 天有效期内，您仍然可以查看免费指标的历史数据。如果您为该控制面板启用了高级指标和建议，则此期限为 15 个月。要访问此数据，您可以在有效期内重新启用控制面板。
- 如果删除控制面板，将丢失所有控制面板的配置设置。您将不再收到任何新的每日指标，而且您也无法访问与该控制面板关联的历史数据。如果要访问已删除控制面板的历史数据，则必须在同一主区域中创建另一个具有相同名称的控制面板。

主题

查看 Amazon S3 Storage Lens 存储统计管理工具控制面板

以下过程显示如何在 S3 控制台中查看 S3 Storage Lens 存储统计管理工具控制面板。请参阅 Amazon S3 Storage Lens 存储统计管理工具使用案例 中基于应用场景的介绍，了解如何使用控制面板优化成本、实施最佳实践以及提高访问 S3 桶的应用程序的性能。

> **Note**
>
> 您不能使用账户的根用户凭证查看 Amazon S3 Storage Lens 存储统计管理工具控制面板。要访问 S3 Storage Lens 存储统计管理工具控制面板，您必须向新的或现有的 IAM 用户授予所需的 AWS Identity and Access Management（IAM）权限。然后，使用这些用户凭证登录以访问

S3 Storage Lens 存储统计管理工具控制面板。有关更多信息，请参阅《IAM 用户指南》中的 Amazon S3 Storage Lens 存储统计管理工具权限 和 IAM 中的安全最佳实践。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要查看的控制面板。

   控制面板在 S3 Storage Lens 存储统计管理工具中打开。Snapshot for date（日期的快照）部分显示 S3 Storage Lens 存储统计管理工具收集指标的最新日期。您的控制面板始终加载指标可用的最新日期。
4. （可选）要更改 S3 Storage Lens 存储统计管理工具控制面板的日期，请在右上角的日期选择器中选择一个新日期。
5. （可选）要应用临时筛选条件以进一步限制控制面板数据的范围，请执行以下操作：

   a. 展开筛选条件部分。

   b. 要按特定账户、AWS 区域、存储类、桶、前缀或 Storage Lens 组进行筛选，请选择相应的筛选依据选项。

   > **Note**
   >
   > 不能同时应用前缀筛选条件和 Storage Lens 组筛选条件。

   c. 要更新筛选条件，请选择 Apply（应用）。

   d. 要移除筛选条件，请单击筛选条件旁的 X。
6. 在 S3 Storage Lens 存储统计管理工具控制面板的任何部分中，要查看特定指标的数据，请为 Metric（指标）选择指标名称。
7. 在 S3 Storage Lens 存储统计管理工具控制面板中的任何图表或可视化效果中，您都可以使用账户、AWS 区域、存储类、存储桶、前缀或 Storage Lens 组选项卡，深入到更深层次的聚合。有关示例，请参阅 发现冷的 Amazon S3 桶。

了解 S3 Storage Lens 存储统计管理工具控制面板

S3 Storage Lens 存储统计管理工具控制面板具有一个主要的 Overview（概览）选项卡，以及表示每个聚合级别的最多五个附加选项卡：

- 账户
- AWS 区域
- 存储类
- 存储桶
- 前缀
- Storage Lens 组

在 Overview（概览）选项卡上，您的控制面板数据聚合为三个不同的部分：Snapshot for date（日期的快照）、Trends and distributions（趋势和分布）以及 Top N overview（前 N 个概览）。

有关 S3 Storage Lens 存储统计管理工具控制面板的更多信息，请参阅以下各节。

快照

Snapshot for date（日期的快照）部分显示 S3 Storage Lens 存储统计管理工具在选定日期聚合的摘要指标。这些摘要指标包括以下指标：

- 总存储 – 使用的存储总量（以字节为单位）。
- 对象计数 – AWS 账户 中的对象总数。
- 平均对象大小 – 对象的平均大小。
- 活动存储桶 – 账户中存储大于 0 字节，处于活动使用状态的活动桶总数。
- 账户 – 其存储在范围内的账户数量。除非您使用的是 AWS Organizations，且您的 S3 Storage Lens 存储统计管理工具具有有效的服务相关角色的可信访问权限，否则此值为 1。有关更多信息，请参阅将服务相关角色用于 Amazon S3 Storage Lens 存储统计管理工具。
- 存储桶 – 账户中的桶总数。

指标数据

对于快照中显示的每个指标，您可以看到以下数据：

- 指标名称 – 指标的名称。
- 指标类别 – 将指标组织到的类别。
- 日期 的总计 – 所选日期的总计数。
- 变化百分比 – 与上次快照日期相比的变化百分比。
- 30 天趋势 – 显示指标在 30 天期间内变化的趋势线。

- 建议 – 基于快照中提供的数据的上下文建议。建议与高级指标和建议一起提供。有关更多信息，请参阅建议。

指标类别

您可以选择更新控制面板 Snapshot for date（日期的快照）部分，以显示其他类别的指标。如果您想查看其他指标的快照数据，可以从以下 Metrics categories（指标类别）中进行选择：

- 成本优化
- 数据保护
- 活动（适用于高级指标）
- 访问管理
- 性能
- 事件

Snapshot for date（日期的快照）部分仅显示每个类别的指标选择。要查看特定类别的所有指标，请在 Trends and distributions（趋势和分布）或 Top N overview（前 N 个概览）部分中选择该指标。有关指标类别的更多信息，请参阅指标类别。要获得 S3 Storage Lens 存储统计管理工具指标的完整列表，请参阅Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

趋势和分布

Overview（概览）选项卡的第二部分是 Trends and distributions（趋势和分布）。在 Trends and distributions（趋势和分布）部分中，您可以选择两个指标以在您定义的日期范围内进行比较。Trends and distributions（趋势和分布）部分显示两个指标在一段时间内的关系。使用此部分显示的图表，您可以查看正在跟踪的两个趋势之间的 Storage class（存储类）和 Region（区域）分布。您可以选择向下钻取其中一个图表中的数据点以进行更深入的分析。

有关使用 Trends and distributions（趋势和分布）部分的演练，请参阅识别不使用具有 AWS KMS 的服务器端加密（SSE-KMS）进行原定设置加密的桶。

前 N 个概述

S3 Storage Lens 存储统计管理工具控制面板的第三部分是前 N 个概述（按升序或降序排序）。此部分将跨最大数量的账户、AWS 区域、桶、前缀或 Storage Lens 组，显示您选择的指标。如果您启用了 S3 Storage Lens 存储统计管理工具以与 AWS Organizations 配合使用，您还可以跨组织查看您选择的指标。

有关使用 Top N overview（前 N 个概览）部分的演练，请参阅识别您最大的 S3 桶。

按选项深入探究和分析

为了提供流畅的分析体验，S3 Storage Lens 存储统计管理工具控制面板提供了一个操作菜单，当您选择任何图表值时将显示该菜单。要使用该菜单，请选择任意图表值以查看关联的指标值，然后在出现的框中从两个选项中进行选择：

- Drill down（深入了解）操作将所选值作为筛选条件应用于控制面板的所有选项卡。然后，您可以深入查看该值以进行更深入的分析。
- 分析依据操作可将您带到您选择的维度选项卡，并将该选项卡值作为筛选条件应用。这些选项卡包括账户、AWS 区域、存储类、存储桶、前缀（适用于启用了高级指标和前缀聚合的控制面板）和 Storage Lens 组（适用于启用了高级指标和 Storage Lens 组聚合的控制面板）。使用分析依据，您可以在新维度的上下文中查看数据以进行更深入的分析。

如果会产生不合逻辑或没有任何价值的结果，则向下钻取和分析依据操作可能会被禁用。向下钻取和分析依据操作都会在控制面板所有选项卡中的任何现有筛选条件之上应用筛选条件。您还可以根据需要移除筛选条件。

选项卡

维度级别选项卡提供了特定维度中所有值的详细视图。例如，AWS 区域选项卡显示所有 AWS 区域的指标，存储桶选项卡显示所有桶的指标。每个维度选项卡包含由四个部分组成的相同布局：

- 趋势图表显示在过去 30 天内所选指标在维度中的前 N 个项目。默认情况下，此图表显示前 10 个项目，但您可以将其减少到至少 3 个项目或将其增加到最多 50 个项目。
- 直方图显示所选日期和指标的垂直条形图。如果您要在此图中显示的项目数量非常大，则可能需要水平滚动。
- 气泡分析图根据维度内的所有项目绘制。此图表用 x 轴表示第一个指标，用 y 轴表示第二个指标，用气泡大小表示第三个指标。
- 指标网格视图包含维度中的每个项目，以行列出。这些列代表每个可用的指标，在指标类别选项卡中排列，以便于导航。

使用数据导出查看 Amazon S3 Storage Lens 存储统计管理工具指标

Amazon S3 Storage Lens 存储统计管理工具指标每天以 CSV 或 Apache Parquet 格式的指标导出文件生成，并放置在您账户的 S3 桶中。您可以将导出的指标从 S3 桶引入自己选择的分析工具中，例如 Amazon QuickSight 和 Amazon Athena，在这里您可以分析存储使用情况和活动趋势。

主题

- 使用 AWS KMS key 加密您的指标导出
- 什么是 S3 Storage Lens 存储统计管理工具导出清单？
- 了解 Amazon S3 Storage Lens 存储统计管理工具导出架构

使用 AWS KMS key 加密您的指标导出

要向 Amazon S3 Storage Lens 存储统计管理工具授予使用客户托管式密钥加密指标导出的权限，必须使用密钥策略。要更新密钥策略，以便您可以使用 KMS 密钥加密 S3 Storage Lens 存储统计管理工具指标导出，请按照以下步骤操作。

授予 S3 Storage Lens 存储统计管理工具使用您的 KMS 密钥加密数据的权限

1. 使用拥有客户托管式密钥的 AWS 账户登录 AWS Management Console。
2. 从 https://console.aws.amazon.com/kms 打开 AWS KMS 控制台。
3. 要更改 AWS 区域，请使用页面右上角的 Region selector (区域选择器)。
4. 在左侧导航窗格中，选择 Customer managed keys (客户托管密钥)。
5. 在客户托管密钥下，选择要用于加密指标导出的密钥。AWS KMS keys 是特定于区域的，必须与指标导出目标 S3 桶位于同一区域中。
6. 在 Key policy (密钥策略) 下，选择 Switch to policy view (切换到策略视图)。
7. 要更新密钥策略，选择 Edit (编辑)。
8. 在 Edit key policy (编辑密钥策略) 下，将以下密钥策略添加到现有密钥策略。要使用这一策略，请将 *user input placeholders* 替换为您的信息。

```
{
    "Sid": "Allow Amazon S3 Storage Lens use of the KMS key",
     "Effect": "Allow",
    "Principal": {
        "Service": "storage-lens.s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
       "StringEquals": {
           "aws:SourceArn": "arn:aws:s3:us-east-1:source-account-id:storage-
lens/your-dashboard-name",
```

```
            "aws:SourceAccount": "source-account-id"
        }
    }
}
```

9. 选择 Save changes（保存更改）。

有关创建客户托管和使用密钥策略的更多信息，请参阅 AWS Key Management Service 开发人员指南中的以下主题：

- 入门
- 使用 AWS KMS 中的密钥策略

您还可以使用 AWS KMS PUT 密钥策略 API 操作（PutKeyPolicy）将密钥策略复制到客户托管式密钥，用于通过 REST API、AWS CLI 和开发工具包对指标导出进行加密。

什么是 S3 Storage Lens 存储统计管理工具导出清单？

鉴于聚合的数据量大，S3 Storage Lens 存储统计管理工具每日指标导出可以拆分为多个文件。清单文件 `manifest.json` 描述了当天的指标导出文件的位置。无论何时交付新的导出，都会附有新的清单。`manifest.json` 文件中包含的每个清单均提供了有关导出的元数据和其他基本信息。

清单信息包括以下属性：

- `sourceAccountId` – 配置拥有者的账户 ID。
- `configId` – 控制面板的唯一标识符。
- `destinationBucket` – 放置指标导出的目标桶 Amazon Resource Name（ARN）。
- `reportVersion` – 导出的版本。
- `reportDate` – 报告的日期。
- `reportFormat` – 报告的格式。
- `reportSchema` – 报告的架构。
- `reportFiles` – 目标桶中的导出报告文件的实际列表。

下面是 CSV 格式导出的 `manifest.json` 文件中的清单示例。

```
{
   "sourceAccountId":"123456789012",
```

```
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"CSV",

 "reportSchema":"version_number,configuration_id,report_date,aws_account_number,aws_region,stor
   "reportFiles":[
        {
            "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-
configuration-id/V_1/reports/dt=2020-11-03/a38f6bc4-2e3d-4355-ac8a-e2fdcf3de158.csv",
            "size":1603959,
            "md5Checksum":"2177e775870def72b8d84febe1ad3574"
        }
      ]
}
```

下面是 Parquet 格式导出的 `manifest.json` 文件中的清单示例。

```
{
   "sourceAccountId":"123456789012",
   "configId":"my-dashboard-configuration-id",
   "destinationBucket":"arn:aws:s3:::destination-bucket",
   "reportVersion":"V_1",
   "reportDate":"2020-11-03",
   "reportFormat":"Parquet",
   "reportSchema":"message s3.storage.lens { required string version_number;
 required string configuration_id; required string report_date; required string
 aws_account_number; required string aws_region; required string storage_class;
 required string record_type; required string record_value; required string
 bucket_name; required string metric_name; required long metric_value; }",
   "reportFiles":[
      {
         "key":"DestinationPrefix/StorageLens/123456789012/my-dashboard-configuration-
id/V_1/reports/dt=2020-11-03/bd23de7c-b46a-4cf4-bcc5-b21aac5be0f5.par",
         "size":14714,
         "md5Checksum":"b5c741ee0251cd99b90b3e8eff50b944"
      }
}
```

您可以将指标导出配置为作为控制面板配置的一部分在 Amazon S3 控制台中生成，也可以使用 Amazon S3 REST API、AWS CLI 和开发工具包进行生成。

了解 Amazon S3 Storage Lens 存储统计管理工具导出架构

下表包含 S3 Storage Lens 存储统计管理工具指标导出架构。

属性名称	数据类型	列名称	说明
VersionNumber	字符串	version_number	正在使用的 S3 Storage Lens 存储统计管理工具指标的版本。
ConfigurationId	字符串	configuration_id	S3 Storage Lens 存储统计管理工具配置的 configuration_id 。
ReportDate	字符串	report_date	跟踪指标的日期。
AwsAccountNumber	字符串	aws_account_number	您的 AWS 账户 数字。
AwsRegion	字符串	aws_region	正在跟踪指标的 AWS 区域。
StorageClass	字符串	storage_class	有问题的桶的存储类别。
RecordType	ENUM	record_type	正在报告的构件类型（账户、桶或前缀）。
RecordValue	字符串	record_value	RecordType 构件的值。

属性名称	数据类型	列名称	说明
			Note record_value 采用 URL 编码。
BucketName	字符串	bucket_name	正在报告的桶的名称。
MetricName	字符串	metric_name	正在报告的指标的名称。
MetricValue	长整型	metric_value	正在报告的指标的值。

S3 Storage Lens 存储统计管理工具指标导出示例

以下是基于此架构导出的 S3 Storage Lens 存储统计管理工具指标的示例。

Note

您可以通过在 record_type 列查找 STORAGE_LENS_GROUP_BUCKET 或 STORAGE_LENS_GROUP_ACCOUNT 值来识别 Storage Lens 组的指标。record_value 列将显示 Storage Lens 组的 Amazon 资源名称 (ARN)，例如，arn:aws:s3:us-east-1:123456789012:storage-lens-group/slg-1。

version_r	configuration_id	report_date	aws_account_number	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			StorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedStorageBytes	20000
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedObjectCount	20
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedStorageBytes	2478828742
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedObjectCount	1598961
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			DeleteMarkerObjectCount	1500
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionStorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	StorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ReplicatedStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ReplicatedObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	EncryptedStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	EncryptedObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	DeleteMarkerObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	CurrentVersionStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	CurrentVersionObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sfo	IncompleteMultipartUploadObjectCount	0

以下是 S3 Storage Lens 存储统计管理工具指标导出示例，其中包含 Storage Lens 组数据。

version_numbe	configuration_id	report_date	aws_account_num	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		StorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUObjectCountOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	StorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUObjectCountOlderThan7Days	0

在 CloudWatch 中监控 S3 Storage Lens 存储统计管理工具指标

您可以将 S3 Storage Lens 存储统计管理工具指标发布到 Amazon CloudWatch，以便在 CloudWatch 控制面板中创建运营状况的统一视图。您还可以使用 CloudWatch 功能（如告警和触发操作、指标数学和异常检测）来监控 S3 Storage Lens 存储统计管理工具指标并采取措施。此外，CloudWatch API 操作使应用程序（包括第三方提供商）能够访问 S3 Storage Lens 存储统计管理工具指标。有关 CloudWatch 功能的更多信息，请参阅 Amazon CloudWatch 用户指南。

您可以使用 Amazon S3 控制台、Amazon S3 REST API、AWS CLI 和 AWS SDK 针对新的或当前控制面板配置启用 CloudWatch 发布选项。升级到 S3 Storage Lens 存储统计管理工具 advanced metrics and recommendations（高级指标和建议）的控制面板可以使用 CloudWatch 发布选项。有关 S3 Storage Lens 存储统计管理工具高级指标和建议定价，请参阅 Amazon S3 定价。不会收取其他 CloudWatch 指标发布费用；但是，其他 CloudWatch 费用（例如控制面板、告警和 API 调用）确实适用。有关更多信息，请参阅 Amazon CloudWatch 定价。

S3 Storage Lens 存储统计管理工具指标在拥有 S3 Storage Lens 存储统计管理工具配置的账户中发布到 CloudWatch。在高级指标和建议中启用 CloudWatch 发布选项后，您可以在 CloudWatch 中访问组织、账户和桶级指标。前缀级别的指标在 CloudWatch 中不可用。

> **Note**
>
> S3 Storage Lens 存储统计管理工具指标是每日指标，每天发布到 CloudWatch 一次。当您在 CloudWatch 中查询 S3 Storage Lens 存储统计管理工具指标时，查询的期限必须为 1 天（86400 秒）。在您的每日 S3 Storage Lens 存储统计管理工具指标显示在 Amazon S3 控制台的 S3 Storage Lens 存储统计管理工具控制面板中后，这些指标可能需要几个小时才能显示在 CloudWatch 中。当您首次为 S3 Storage Lens 存储统计管理工具指标启用 CloudWatch 发布选项时，您的指标最多可能需要 24 小时才能将指标发布到 CloudWatch。

启用 CloudWatch 发布选项后，您可以使用以下 CloudWatch 功能监控和分析 S3 Storage lens 存储统计管理工具数据：

- Dashboards（控制面板）— 使用 CloudWatch 控制面板创建定制 S3 Storage Lens 存储统计管理工具控制面板。将您的 CloudWatch 控制面板与团队间无法直接访问您的 AWS 账户的人员、利益攸关方和组织外部的人员共享。
- Alarms and triggered actions（告警和触发操作）— 配置监视指标的警报，并在超出阈值时采取行动。例如，您可以配置告警，当 Incomplete Multipart Upload Bytes（未完成分段上传字节）指标连续三天超过 1GB 时发送 Amazon SNS 通知。
- Anomaly detection（异常检测）— 启用异常检测以持续分析指标、确定正常基线和表面异常。您可以根据指标的预期值创建异常检测告警。例如，您可以监视 Object Lock Enabled Bytes（启用对象锁定的字节）指标的异常情况，以检测未经授权移除对象锁定设置的情况。
- Metric math（指标数学）– 您还可以使用指标数学查询多个 S3 Storage Lens 存储统计管理工具指标，并使用数学表达式基于这些度量创建新的时间序列。例如，您可以通过将 `StorageBytes` 除以 `ObjectCount` 创建一个新指标来获得平均对象大小。

有关 S3 Storage Lens 存储统计管理工具指标的 CloudWatch 发布选项的更多信息，请参阅以下主题。

主题

- S3 Storage Lens 存储统计管理工具指标和维度
- 为 S3 Storage Lens 存储统计管理工具启用 CloudWatch 发布
- 在 CloudWatch 中结合使用 S3 Storage Lens 存储统计管理工具指标

S3 Storage Lens 存储统计管理工具指标和维度

要将 S3 Storage Lens 存储统计管理工具指标发送到 CloudWatch，您必须在 S3 Storage Lens 存储统计管理工具高级指标和建议中启用 CloudWatch 发布选项。启用高级指标后，您可以使用 CloudWatch 控制面板与其他应用程序指标一起监控 S3 Storage Lens 存储统计管理工具指标，并创建运营状况的统一视图。您可以使用维度在 CloudWatch 中按企业、账户、桶、存储类别、区域和指标配置 ID 筛选 CloudWatch 中的 S3 Storage Lens 存储统计管理工具指标。

S3 Storage Lens 存储统计管理工具指标在拥有 S3 Storage Lens 存储统计管理工具配置的账户中发布到 CloudWatch。在高级指标和建议中启用 CloudWatch 发布选项后，您可以在 CloudWatch 中访问组织、账户和桶级指标。前缀级别的指标在 CloudWatch 中不可用。

> **Note**
>
> S3 Storage Lens 存储统计管理工具指标是每日指标，每天发布到 CloudWatch 一次。当您在 CloudWatch 中查询 S3 Storage Lens 存储统计管理工具指标时，查询的期限必须为 1 天（86400 秒）。在您的每日 S3 Storage Lens 存储统计管理工具指标显示在 Amazon S3 控制台的 S3 Storage Lens 存储统计管理工具控制面板中后，这些指标可能需要几个小时才能显示在 CloudWatch 中。当您首次为 S3 Storage Lens 存储统计管理工具指标启用 CloudWatch 发布选项时，您的指标最多可能需要 24 小时才能将指标发布到 CloudWatch。

有关 CloudWatch 中 S3 Storage Lens 存储统计管理工具指标和维度的更多信息，请参阅以下主题。

主题

- 指标
- 维度

指标

S3 Storage Lens 存储统计管理工具指标在 CloudWatch 中作为指标提供。S3 Storage Lens 存储统计管理工具指标已发布到 AWS/S3/Storage-Lens 命名空间。此命名空间仅适用于 S3 Storage Lens 存储统计管理工具指标。Amazon S3 桶、请求和复制指标已发布到 AWS/S3 命名空间。

S3 Storage Lens 存储统计管理工具指标在拥有 S3 Storage Lens 存储统计管理工具配置的账户中发布到 CloudWatch。在高级指标和建议中启用 CloudWatch 发布选项后，您可以在 CloudWatch 中访问组织、账户和桶级指标。前缀级别的指标在 CloudWatch 中不可用。

在 S3 Storage Lens 存储统计管理工具中，指标会聚合并仅存储在指定的主区域中。S3 Storage Lens 存储统计管理工具指标也发布到您在 S3 Storage Lens 存储统计管理工具配置中指定的主页区域中的 CloudWatch。

有关 S3 Storage Lens 存储统计管理工具指标的完整列表，包括 CloudWatch 中可用的指标列表，请参阅 Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

> **Note**
>
> CloudWatch 中 S3 Storage Lens 存储统计管理工具指标的有效统计数据是平均值。有关 CloudWatch 中统计信息的更多信息，请参阅 Amazon CloudWatch 用户指南中的 CloudWatch 统计数据定义。

CloudWatch 中 S3 Storage Lens 存储统计管理工具指标的粒度

S3 Storage Lens 存储统计管理工具提供企业、账户、桶和前缀粒度的指标。S3 Storage Lens 存储统计管理工具将企业、账户和桶级 S3 Storage Lens 存储统计管理工具指标发布到 CloudWatch。前缀级别的 S3 Storage Lens 存储统计管理工具指标在 CloudWatch 中不可用。

有关 CloudWatch 中可用 S3 Storage Lens 存储统计管理工具指标的粒度的更多信息，请参阅以下列表：

- Organization（企业）— 在企业中的成员账户中聚合的指标。S3 Storage Lens 存储统计管理工具将成员账户的指标发布到管理账户中的 CloudWatch。
  - Organization and account（企业和账户）— 在企业中的成员账户指标。
  - Organization and bucket（企业和桶）— 在企业的成员账户中的 Amazon S3 桶指标。
- 账户（非企业级别）— 您账户中桶之间汇总的指标。

- 桶（非企业级别）— 特定桶的指标。在 CloudWatch 中，S3 Storage Lens 存储统计管理工具将这些指标发布至创建 S3 Storage Lens 存储统计管理工具配置的 AWS 账户。S3 Storage Lens 存储统计管理工具仅针对非企业配置发布这些指标。

维度

当 S3 Storage Lens 存储统计管理工具向 CloudWatch 发送数据时，维度附加到每个指标。维度是描述指标特征的类别。您可使用维度筛选 CloudWatch 返回的结果。

例如，CloudWatch 中的所有 S3 Storage Lens 存储统计管理工具指标都具有 `configuration_id` 维度。您可以使用此维度来区分与特定 S3 Storage Lens 存储统计管理工具配置关联的指标。`organization_id` 识别企业级指标。有关维度的更多信息，请参阅 CloudWatch User Guide（CloudWatch 用户指南）中的 Dimensions（维度）。

S3 Storage Lens 存储统计管理工具指标有不同的维度，具体取决于指标的粒度。例如，您可以使用 `organization_id` 维度按照 AWS Organizations ID 筛选企业级的指标。但是，您不能将此维度用于桶和账户级指标。有关更多信息，请参阅使用维度筛选指标。

要查看哪些维度可用于 S3 Storage Lens 存储统计管理工具配置，请参阅下表。

维度	描述	桶	账户	organization（组织）	组织和桶	组织和账户
`configuration_id`	指标中报告的 S3 Storage Lens 存储统计管理工具配置的控制面板名称					
`metrics_version`	S3 Storage Lens 存储统计管理工具指标的版本。指标版本的固定值为 `1.0`。					
`organization_id`	指标的 AWS Organizations ID					
`aws_account_number`	与指标关联的 AWS 账户					

维度	描述	桶	账户	organization（组织）	组织和桶	组织和账户
aws_region	指标的 AWS 区域	.	.	.	.	.
bucket_name	指标中报告的 S3 桶的名称	.	.	.	.	.
storage_class	指标中报告的桶的存储类	.	.	.	.	.
record_type	指标的粒度：组织、账户、桶	桶	账户	桶	账户	企业

为 S3 Storage Lens 存储统计管理工具启用 CloudWatch 发布

您可以将 S3 Storage Lens 存储统计管理工具指标发布到 Amazon CloudWatch，以便在 CloudWatch 控制面板中创建运营状况的统一视图。您还可以使用 CloudWatch 功能（如告警和触发操作、指标数学和异常检测）来监控 S3 Storage Lens 存储统计管理工具指标并采取措施。此外，CloudWatch API 操作使应用程序（包括第三方提供商）能够访问 S3 Storage Lens 存储统计管理工具指标。有关 CloudWatch 功能的更多信息，请参阅 Amazon CloudWatch 用户指南。

S3 Storage Lens 存储统计管理工具指标在拥有 S3 Storage Lens 存储统计管理工具配置的账户中发布到 CloudWatch。在高级指标和建议中启用 CloudWatch 发布选项后，您可以在 CloudWatch 中访问组织、账户和桶级指标。前缀级别的指标在 CloudWatch 中不可用。

您可以使用 Amazon S3 控制台、REST API、AWS CLI 和 AWS SDK 针对新的或当前控制面板配置启用 CloudWatch 支持。CloudWatch 发布选项可用于升级到 S3 Storage Lens 存储统计管理工具高级指标和建议。有关 S3 Storage Lens 存储统计管理工具高级指标和建议定价，请参阅 Amazon S3 定价。不会收取其他 CloudWatch 指标发布费用；但是，其他 CloudWatch 费用（例如控制面板、告警和 API 调用）确实适用。

要启用 S3 Storage Lens 存储统计管理工具指标的 CloudWatch 发布选项，请参阅以下主题。

> **Note**
>
> S3 Storage Lens 存储统计管理工具指标是每日指标，每天发布到 CloudWatch 一次。当您在 CloudWatch 中查询 S3 Storage Lens 存储统计管理工具指标时，查询的期限必须为 1 天（86400 秒）。在您的每日 S3 Storage Lens 存储统计管理工具指标显示在 Amazon S3 控制台的 S3 Storage Lens 存储统计管理工具控制面板中后，这些指标可能需要几个小时才能显示在 CloudWatch 中。当您首次为 S3 Storage Lens 存储统计管理工具指标启用 CloudWatch 发布选项时，您的指标最多可能需要 24 小时才能将指标发布到 CloudWatch。
> 目前，S3 Storage Lens 存储统计管理工具指标无法通过 CloudWatch 流使用。

使用 S3 控制台

更新 S3 Storage Lens 存储统计管理工具控制面板时，无法更改控制面板名称或主区域。您也不能更改原定设置控制面板的范围，其范围限于整个账户的存储。

更新 S3 Storage Lens 存储统计管理工具控制面板以启用 CloudWatch 发布

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，依次选择 S3 Storage Lens 和 Dashboards（控制面板）。
3. 请选择要编辑的控制面板，然后选择 Edit（编辑）。
4. 在 Metrics selection（指标选择）下方，请选择 Advanced metrics and recommendations（高级指标和建议）。

   高级指标和建议可额外收费。高级指标和建议包括 15 个月的数据查询期、在前缀级别聚合的使用情况指标、按桶聚合的活动指标、CloudWatch 发布选项，以及帮助您优化存储成本和应用数据保护最佳做法的上下文建议。有关更多信息，请参阅 Amazon S3 定价。
5. 在 Select Advanced metrics and recommendations features（选择高级指标和推荐功能）下方，选择 CloudWatch publishing（CloudWatch 发布）。

   > **Important**
   >
   > 如果您的配置启用了使用情况指标的前缀聚合，则前缀级指标将不会发布到 CloudWatch。只有桶、账户和企业级 S3 Storage Lens 存储统计管理工具指标才会发布到 CloudWatch。
6. 选择 Save changes（保存更改）。

要创建启用 CloudWatch 支持的新 S3 Storage Lens 存储统计管理工具控制面板

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，依次选择 Storage Lens 和 Dashboards（控制面板）。
3. 请选择创建控制面板。
4. 在 General（常规）下，定义以下配置选项：

   a. 对于 Dashboard name（控制面板名称），输入您的控制面板名称。

      控制面板名称必须少于 65 个字符，且不得包含特殊字符或空格。创建控制面板后，您无法更改此控制面板名称。

   b. 为控制面板选择 Home Region（主区域）。

      此控制面板范围内所有包含的所有区域的指标，都集中存储在此指定的主区域中。在 CloudWatch 中，S3 Storage Lens 存储统计管理工具指标也可在主区域提供。创建控制面板后，您无法更改主区域。

5. （可选）要添加标签，请选择 Add tag （添加标签），然后输入标签 Key（密钥）和 Value（值）。

   > **Note**
   >
   > 您最多可以在控制面板配置中添加 50 个标签。

6. 定义配置的范围：

   a. 如果您正在创建组织级配置，请选择要包含在配置中的账户：Include all accounts in your configuration（在配置中包括所有账户）或者 Limit the scope to your signed-in account（将范围限制为您的登录账户）。

      > **Note**
      >
      > 当您创建包含所有账户的组织级配置时，您只能包括或排除区域，而不能包括或排除桶。

   b. 通过执行以下操作，选择您希望 S3 Storage Lens 存储统计管理工具在控制面板配置中包括的区域和桶：

- 要包含所有区域，请选择 Include Regions and buckets（包含区域和桶）。
- 要包含特定的区域，请清除 Include all Regions（包含所有区域）。在 Choose Regions to include（选择要包含的区域）下方，请选择想要 S3 Storage Lens 存储统计管理工具包含在控制面板中的区域。
- 要包含特定桶，请清除 Include all buckets（包含所有桶）。在 Choose buckets to include（选择要包含的桶）下方，请选择想要 S3 Storage Lens 存储统计管理工具包含在控制面板中的桶。

> **Note**
>
> 您最多可以选择 50 个桶。

7. 对于 Metrics selection（指标选择），请选择 Advanced metrics and recommendations（高级指标和建议）。

   有关高级指标和建议定价的资讯，请参阅 Amazon S3 pricing（Amazon S3 定价）。

8. 在 Advanced metrics and recommendations features（高级指标和建议功能）下方，选择您要启用的选项：

   - Advanced metrics（高级指标）
   - CloudWatch publishing（CloudWatch 发布）

   > **Important**
   >
   > 如果您为 S3 Storage Lens 存储统计管理工具配置启用了前缀聚合，那么前缀级别的指标将不会发布到 CloudWatch。只有桶、账户和企业级 S3 Storage Lens 存储统计管理工具指标才会发布到 CloudWatch。

   - Prefix aggregation（前缀聚合）

   > **Note**
   >
   > 有关高级指标和建议功能的更多信息，请参阅指标选择。

9. 如果您启用了 Advanced metrics（高级指标），请选择要在 S3 Storage Lens 存储统计管理工具控制面板中显示的 Advanced metrics categories（高级指标类别）：

- 活动指标
- Detailed status code metrics（详细的状态代码指标）
- Advanced cost optimization metrics（高级成本优化指标）
- Advanced data protection metrics（高级数据保护指标）

有关指标类别的更多信息，请参阅指标类别。要获得指标的完整列表，请参阅 Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

10. （可选）配置您的指标导出。

    有关如何配置指标导出的更多信息，请参阅步骤 创建 Amazon S3 Storage Lens 存储统计管理工具控制面板。

11. 请选择创建控制面板。

使用 AWS CLI

以下 AWS CLI 示例将通过使用 S3 Storage Lens 存储统计管理工具组织级高级指标和建议配置启用 CloudWatch 发布选项。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control put-storage-lens-configuration --account-id=555555555555 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json

config.json
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
```

```
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "ActivityMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true //Mark this as false if you want only free metrics.
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true, //Mark this as false if you want only free metrics.
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
configurations.
      "arn:aws:s3:::source_bucket1"
    ]
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
```

```
      "Arn": "arn:aws:s3:::destination-bucket-name", // The destination bucket for your
 metrics export must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true //Mark this as false if you want to export only free metrics.
    }
  }
}
```

使用适用于 Java 的 AWS 开发工具包

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;
```

```
import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
 DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
 DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);
```

```
            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
PutStorageLensConfigurationRequest()
```

```
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 REST API

要使用 Amazon S3 REST API 启用 CloudWatch 发布选项，您可以使用 PutStorageLensConfiguration。

后续步骤

启用 CloudWatch 发布选项后，您可以在 CloudWatch 中访问 S3 Storage Lens 存储统计管理工具指标。您还可以利用 CloudWatch 功能在 CloudWatch 中监控和分析 S3 Storage Lens 存储统计管理工具数据。有关更多信息，请参阅以下主题：

- S3 Storage Lens 存储统计管理工具指标和维度
- 在 CloudWatch 中结合使用 S3 Storage Lens 存储统计管理工具指标

在 CloudWatch 中结合使用 S3 Storage Lens 存储统计管理工具指标

您可以将 S3 Storage Lens 存储统计管理工具指标发布到 Amazon CloudWatch，以便在 CloudWatch 控制面板中创建运营状况的统一视图。您还可以使用 CloudWatch 功能（如告警和触发操作、指标数学和异常检测）来监控 S3 Storage Lens 存储统计管理工具指标并采取措施。此外，CloudWatch API 操作使应用程序（包括第三方提供商）能够访问 S3 Storage Lens 存储统计管理工具指标。有关 CloudWatch 功能的更多信息，请参阅 Amazon CloudWatch 用户指南。

您可以使用 Amazon S3 控制台、Amazon S3 REST API、AWS CLI 和 AWS SDK 针对新的或当前控制面板配置启用 CloudWatch 发布选项。CloudWatch 发布选项可用于升级到 S3 Storage Lens 存储统

计管理工具高级指标和建议。有关 S3 Storage Lens 存储统计管理工具高级指标和建议定价，请参阅 Amazon S3 定价。不会收取其他 CloudWatch 指标发布费用；但是，其他 CloudWatch 费用（例如控制面板、告警和 API 调用）确实适用。有关更多信息，请参阅 Amazon CloudWatch 定价。

S3 Storage Lens 存储统计管理工具指标在拥有 S3 Storage Lens 存储统计管理工具配置的账户中发布到 CloudWatch。在高级指标和建议中启用 CloudWatch 发布选项后，您可以在 CloudWatch 中访问组织、账户和桶级指标。前缀级别的指标在 CloudWatch 中不可用。

> **Note**
>
> S3 Storage Lens 存储统计管理工具指标是每日指标，每天发布到 CloudWatch 一次。当您在 CloudWatch 中查询 S3 Storage Lens 存储统计管理工具指标时，查询的期限必须为 1 天（86400 秒）。在您的每日 S3 Storage Lens 指标显示在 Amazon S3 控制台的 S3 Storage Lens 存储统计管理工具控制面板中后，这些指标可能需要几个小时才能显示在 CloudWatch 中。当您首次为 S3 Storage Lens 存储统计管理工具指标启用 CloudWatch 发布选项时，您的指标最多可能需要 24 小时才能将指标发布到 CloudWatch。
> 目前，S3 Storage Lens 存储统计管理工具指标无法通过 CloudWatch 流使用。

有关在 CloudWatch 中使用 S3 Storage Lens 存储统计管理工具指标的更多信息，请参阅以下主题。

主题

结合使用 CloudWatch 控制面板

您可以使用 CloudWatch dashboards（控制面板）与其他应用程序指标一起监控 S3 Storage Lens 存储统计管理工具指标，并创建运营状况的统一视图。控制面板是 CloudWatch 控制台中的一种自定义主页，可用于在单一视图中监视资源。

CloudWatch 具有广泛的权限控制，不支持限制对一组特定指标或维度的访问。您的账户或企业中有权访问 CloudWatch 的用户将有权访问启用了 CloudWatch 支持选项的所有 S3 Storage Lens 存储统计管理工具配置的指标。您无法像在 S3 Storage Lens 存储统计管理工具中那样管理特定控制面

板的权限。有关 CloudWatch 权限的更多信息，请参阅 Amazon CloudWatch User Guide（Amazon CloudWatch 用户指南）中的 Managing access permissions to your CloudWatch resources（管理对您的 CloudWatch 资源的访问权限）。

有关使用 CloudWatch 控制面板和配置权限的更多信息，请参阅 Amazon CloudWatch User Guide（Amazon CloudWatch 用户指南）中的 Using Amazon CloudWatch dashboards（使用 Amazon CloudWatch 控制面板）和 Sharing CloudWatch dashboards（共享 CloudWatch 控制面板）。

设置警报、触发操作和使用异常检测

您可以配置 CloudWatch 警报来监视 CloudWatch 中的 S3 Storage Lens 存储统计管理工具指标，并在超出阈值时采取行动。例如，您可以配置告警，当 Incomplete Multipart Upload Bytes（未完成分段上传字节）指标连续三天超过 1GB 时发送 Amazon SNS 通知。

您还可以启用异常检测以持续分析 S3 Storage Lens 存储统计管理工具指标、确定正常基线和表面异常。您可以根据指标的预期值创建异常检测告警。例如，您可以监视 Object Lock Enabled Bytes（启用对象锁定的字节）指标的异常情况，以检测未经授权移除对象锁定设置的情况。

有关更多信息以及示例，请参阅 Amazon CloudWatch User Guide（Amazon CloudWatch 用户指南）中的 Using Amazon CloudWatch alarms（使用 Using Amazon CloudWatch 告警）和 Creating an alarm from a metric on a graph（从图表上的指标创建告警）。

使用维度筛选指标

您可以在 CloudWatch 控制台中使用维度筛选 S3 Storage Lens 存储统计管理工具指标。例如，您可以按照 `configuration_id`、`aws_account_number`、`aws_region`、`bucket_name` 等方式进行筛选。

S3 Storage Lens 存储统计管理工具支持每个账户多个控制面板配置。这意味着不同的配置可以包含同一个桶。当这些指标发布到 CloudWatch 时，桶将在 CloudWatch 中具有重复的指标。要是只需查看 CloudWatch 中特定 S3 Storage Lens 存储统计管理工具配置的指标，您可以使用 `configuration_id` 维度。当您按照 `configuration_id` 筛选时，您只能看到与您标识的配置相关联的指标。

有关按配置 ID 进行筛选的更多信息，请参阅《Amazon CloudWatch 用户指南》中的搜索可用指标。

使用指标数学计算新指标

您可以使用指标数学查询多个 S3 Storage Lens 存储统计管理工具指标，并使用数学表达式基于这些度量创建新的时间序列。例如，您可以通过从对象计数中减去加密对象来为未加密对象创建一个新的指

标。您还可以创建一个指标来获得平均对象大小，方法是将 StorageBytes 除以 ObjectCount，或者通过将 BytesDownloaded 除以 StorageBytes 来获得一天访问的字节数。

有关更多信息，请参阅 Amazon CloudWatch User Guide（Amazon CloudWatch 用户指南）中的 Using metric math（使用指标数学）。

在图表中使用搜索表达式

借助 S3 Storage Lens 存储统计管理工具指标，您可以创建搜索表达式。例如，您可以为所有名为 IncompleteMultipartUploadStorageBytes 的指标创建搜索表达式，并将 SUM 添加至表达式。使用此搜索表达式，您可以在单个指标中跨存储的所有维度查看未完成分段上传总字节数。

这个例子展示了用来为所有名为 IncompleteMultipartUploadStorageBytes 的指标创建搜索表达式的语法。

```
SUM(SEARCH('{AWS/S3/Storage-
Lens,aws_account_number,aws_region,configuration_id,metrics_version,record_type,storage_class}
 MetricName="IncompleteMultipartUploadStorageBytes"', 'Average',86400))
```

有关此语法的更多信息，请参阅 Amazon CloudWatch User Guide（Amazon CloudWatch 用户指南）中的 CloudWatch search expression syntax（CloudWatch 搜索表达式语法）。要使用搜索表达式创建 CloudWatch 图表，请参阅 Amazon CloudWatch User Guide（Amazon CloudWatch 用户指南）中的 Creating a CloudWatch graph with a search expression（使用搜索表达式创建 CloudWatch 图表）。

Amazon S3 Storage Lens 存储统计管理工具使用案例

您可以使用 Amazon S3 Storage Lens 存储统计管理工具控制面板直观显示见解和趋势，标记异常值，并接收建议。S3 Storage Lens 存储统计管理工具指标分为与关键使用案例一致的类别。您可以使用这些指标来进行以下操作：

- 识别成本优化机会
- 应用数据保护最佳实践
- 应用访问管理最佳实践
- 提高应用程序工作负载的性能

例如，使用成本优化指标，您可以识别可降低您的 Amazon S3 存储成本的机会。您可以识别分段上传时间超过 7 天的桶，也可以识别正在累积非当前版本的桶。

同样，您可以使用数据保护指标来识别组织内未遵循数据保护最佳实践的桶。例如，您可以识别未使用 AWS Key Management Service 密钥（SSE-KMS）进行原定设置加密或未启用 S3 版本控制的桶。

使用 S3 Storage Lens 存储统计管理工具访问管理指标，您可以识别 S3 对象所有权的桶设置，这样您就可以将访问控制列表（ACL）权限迁移到桶策略并禁用 ACL。

如果启用了 S3 Storage Lens 存储统计管理工具高级指标，您可以使用详细的状态代码指标了解成功或失败的请求的计数，这些计数可用于解决访问或性能问题。

借助高级指标，您还可以访问更多的成本优化和数据保护指标，这些指标可用于发现进一步降低整体 S3 存储成本的机会，并更好地遵循保护数据的最佳实践。例如，高级成本优化指标包括生命周期规则计数，您可以使用生命周期规则计数来识别没有生命周期规则的桶，从而使超过 7 天的未完成分段上传过期。高级数据保护指标包括复制规则计数。

有关指标类别的更多信息，请参阅指标类别。要获得 S3 Storage Lens 存储统计管理工具指标的完整列表，请参阅Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

主题

- 使用 Amazon S3 Storage Lens 存储统计管理工具优化您的存储成本
- 使用 S3 Storage Lens 存储统计管理工具保护您的数据
- 使用 S3 Storage Lens 存储统计管理工具审计对象所有权设置
- 使用 S3 Storage Lens 存储统计管理工具指标提高性能

使用 Amazon S3 Storage Lens 存储统计管理工具优化您的存储成本

您可以使用 S3 Storage Lens 存储统计管理工具成本优化指标来降低 S3 存储的总体成本。成本优化指标可以帮助您确认您已根据最佳实践经济高效地配置 Amazon S3。例如，您可以确定以下成本优化机会：

- 具有超过 7 天的未完成分段上传的桶
- 累积了许多非当前版本的桶
- 不具有生命周期规则来中止未完成分段上传的桶
- 不具有使非当前版本对象过期的生命周期规则的桶
- 不具有将对象转换为其他存储类的生命周期规则的桶

然后，您可以使用这些数据向桶添加额外的生命周期规则。

以下示例说明如何使用 S3 Storage Lens 存储统计管理工具控制面板中的成本优化指标来优化存储成本。

主题

识别您最大的 S3 桶

您要为在 S3 桶中存储对象付费。您需付费的费率取决于对象的大小、对象的存储时长及其存储类。利用 S3 Storage Lens 存储统计管理工具，您可以集中查看账户中的所有桶。要查看组织的所有账户中的所有桶，您可以配置 AWS Organizations 级 S3 Storage Lens 存储统计管理工具控制面板。在此控制面板视图中，您可以识别最大的桶。

步骤 1：识别最大的 S3 桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要查看的控制面板。

   控制面板打开后，您可以看到 S3 Storage Lens 存储统计管理工具收集指标的最新日期。您的控制面板始终加载指标可用的最新日期。
4. 要查看所选日期范围内按 Total storage（总存储）指标排列的最大桶的排名，请向下滚动到 Top N overview for date（日期的前 N 个概览）部分。

   您可以切换排序顺序以显示最小的桶。您还可以调整 Metric（指标）选择，以根据任何可用指标对桶进行排名。Top N overview for date（日期的前 N 个概览）部分还显示与前一天或前一周相比的百分比变化以及用于可视化趋势的火花线。该趋势对于免费指标为 14 天趋势，对于高级指标和建议为 30 天趋势。

> **Note**
>
> 使用 S3 Storage Lens 存储统计管理工具高级指标和建议，指标可用于 15 个月的查询。有关更多信息，请参阅指标选择。

5. 要了解有关您的桶的更详细的见解，请向上滚动到页面顶部，然后选择 Bucket（桶）选项卡。

   在 Bucket（桶）选项卡上，您可以看到诸如近期增长率、平均对象大小、最大前缀和对象数等详细信息。

步骤 2：导航到您的桶并进行调查

在确定最大的 S3 桶之后，您可以导航到 S3 控制台中的每个桶，以查看桶中的对象，了解其关联的工作负载，或识别桶的内部拥有者。您可以与桶拥有者联系，以了解此增长是否符合预期，或者此增长是否需要进一步监控和控制。

发现冷的 Amazon S3 桶

如果启用了 S3 Storage Lens 存储统计管理工具高级指标，您可以使用活动指标了解 S3 桶的活跃度。“冷”桶是指不再访问（或极少访问）其存储的桶。这种活跃程度缺乏通常表示桶的对象不经常访问。

活动指标，例如 GET Requests（Get 请求）和 Download Bytes（下载字节数），指明每天访问桶的频率。要了解访问模式的一致性并发现根本不再访问的桶，您可以对这些数据进行几个月的趋势分析。Retrieval rate（检索率）指标（计算方法为下载字节数/总存储），表示桶中每天访问的存储比例。

> **Note**
>
> 如果同一对象在一天中多次下载，则下载字节数会加倍。

先决条件

要在 S3 Storage Lens 存储统计管理工具控制面板中查看活动指标，您必须启用 S3 Storage Lens 存储统计管理工具 Advanced metrics and recommendations（高级指标和建议），然后选择 Activity metrics（活动指标）。有关更多信息，请参阅创建和更新 Amazon S3 Storage Lens 存储统计管理工具控制面板。

步骤 1：识别活动的桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要查看的控制面板。
4. 选择 Bucket（桶）选项卡，然后向下滚动到 Bubble analysis by buckets for date（按日期的桶划分的气泡分析）图表。

   在 Bubble analysis by buckets for date（按日期的桶划分的气泡分析）部分中，您可以使用任意三个指标在多个维度上绘制桶，以表示气泡的 X-axis（X 轴）、Y-axis（Y 轴）和 Size（大小）。
5. 要查找变冷的桶，请为 X-axis（X 轴）、Y-axis（Y 轴）和 Size（大小）选择 Total storage（总存储空间）、% retrieval rate（检索率百分比）和 Average object size（平均对象大小）指标。
6. 在 Bubble analysis by buckets for date（按日期的桶划分的气泡分析）部分中，查找检索率为零（或接近零）且相对存储大小较大的任何桶，然后选择代表桶的气泡。

   此时将出现一个方框，其中包含可提供更精细见解的选项。请执行下列操作之一：

   a. 要更新 Bucket（桶）选项卡以仅显示选定桶的指标，请选择 Drill down（深入分析），然后选择 Apply（应用）。
   b. 要按账户、AWS 区域、存储类或桶聚合桶级别数据，请选择 Analyze by（分析依据），然后选择 Dimension（维度）。例如，要按存储类进行聚合，请为 Dimension（维度）选择 Storage class（存储类）。

   要查找已经冷的桶，请使用 Total storage（总存储）、% retrieval rate（检索率百分比）和 Average object size（平均对象大小）指标执行气泡分析。查找检索率为零（或接近零）且相对存储大小较大的任何桶。

   控制面板的 Bucket（桶）选项卡将更新，以显示所选聚合或筛选条件的数据。如果您按存储类或其他维度进行汇总，则该新选项卡将在您的控制面板中打开 [例如，Storage class（存储类）选项卡]。

步骤 2：调查冷桶

在此处，您可以识别您账户或组织中冷桶的拥有者，并了解是否仍然需要该存储。然后，您可以通过为这些桶配置生命周期过期配置或将数据存档到其中一个 Amazon S3 Glacier 存储类来优化成本。

为了避免冷桶继续发展的问题，您可以使用 S3 生命周期配置为桶自动转换数据，也可以启用使用 S3 Intelligent-Tiering 自动存档。

您也可以使用步骤 1 来识别热桶。然后，您可以确保这些桶使用正确的 S3 存储类，以确保它们在性能和成本方面最有效地处理请求。

查找未完成的分段上传

您可以使用分段上传将非常大的对象（最大 5TB）分成多个分段上载，以提高吞吐量并更快地从网络问题中恢复。如果分段上传过程未完成，则未完成的分段将保留在桶中（处于不可用状态）。这些未完成的分段将产生存储成本，直到上载过程完成或删除未完成的分段。有关更多信息，请参阅使用分段上传来上传和复制对象。

借助 S3 Storage Lens 存储统计管理工具，您可以确定账户中或整个组织中未完成的分段上传字节数，包括超过 7 天的未完成多段上传。有关未完成分段上传指标的完整列表，请参阅Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

作为最佳实践，我们建议配置生命周期规则，使超过特定天数的未完成分段上传过期。当您创建生命周期规则以使未完成分段上传过期时，我们建议以 7 天作为一个良好的起点。

步骤 1：查看未完成分段上传的总体趋势

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要查看的控制面板。
4. 在 Snapshot for date（日期的快照）部分中的 Metrics categories（指标类别）下，选择 Cost optimization（成本优化）。

   Snapshot for date（日期的快照）部分将更新，以显示 Cost optimization（成本优化）指标，其中包括 Incomplete multipart upload bytes greater than 7 days old（超过 7 天的未完成分段上传字节数）。

   在 S3 Storage Lens 存储统计管理工具控制面板的任何图表中，您都可以看到未完成分段上传的指标。您可以使用这些指标进一步评估未完成分段上传字节对存储的影响，包括对总体增长趋势的贡献。您还可以使用 Account（账户）、AWS 区域、Bucket（桶）或 Storage class（存储类）选项卡深入到更深层次的聚合，对数据进行更深入的分析。有关示例，请参阅 发现冷的 Amazon S3 桶。

步骤 2：识别具有最多不完整分段上传字节但没有生命周期规则来中止不完整分段上传的桶

先决条件

要在 S3 Storage Lens 存储统计管理工具控制面板中查看 Abort incomplete multipart upload lifecycle rule count（中止未完成分段上传生命周期规则计数）指标，您必须启用 S3 Storage Lens 存储统计管理工具 Advanced metrics and recommendations（高级指标和建议），然后选择 Advanced cost optimization metrics（高级成本优化指标）。有关更多信息，请参阅创建和更新 Amazon S3 Storage Lens 存储统计管理工具控制面板。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要查看的控制面板。
4. 要确定累积了超过 7 天的未完成分段上传的特定桶，请转到 Top N overview for date（日期的前 N 个概览）部分。

   原定设置情况下，Top N overview for date（日期的前 N 个概览）部分显示前 3 个桶的指标。您可以增加或减少 Top N（前 N 个）字段中的桶数量。Top N overview for date（日期的前 N 个概览）部分还显示与前一天或前一周相比的百分比变化以及用于可视化趋势的火花线。（该趋势对于免费指标为 14 天趋势，对于高级指标和建议为 30 天趋势。）

   > **Note**
   >
   > 使用 S3 Storage Lens 存储统计管理工具高级指标和建议，指标可用于 15 个月的查询。有关更多信息，请参阅指标选择。

5. 对于 Metric（指标），在 Cost optimization（成本优化）类别中选择 Incomplete multipart upload bytes greater than 7 days old（超过 7 天的未完成分段上传字节数）。

   在 Top number buckets（前 number 个桶）下，您可以看到具有最多的超过 7 天的不完整分段上传存储字节的桶。
6. 要查看未完成分段上传的更详细的桶级指标，请滚动到页面顶部，然后选择 Bucket（桶）选项卡。
7. 向下滚动到 Bucket（桶）部分。对于 Metrics categories（指标类别），选择 Cost optimization（成本优化）。然后清除 Summary（摘要）。

   Buckets（桶）列表将更新，以显示所显示的桶的所有可用 Cost optimization（成本优化）指标。

8. 要筛选 Buckets（桶）列表以仅显示特定的成本优化指标，请选择首选项图标（  ）。
9. 清除所有成本优化指标的切换开关，直到只有 Incomplete multipart upload bytes greater than 7 days old（超过 7 天的未完成分段上传字节）和 Abort incomplete multipart upload lifecycle rule count（中止未完成的分段上传生命周期规则计数）保持选中状态。
10. （可选）在 Page size（页面大小）下，选择要在列表中显示的桶数。
11. 选择 Confirm（确认）。

    Buckets（桶）列表会更新，以显示未完成分段上传和生命周期规则计数的桶级指标。您可以使用这些数据来识别具有最多的超过 7 天的不完整分段上传字节且缺少生命周期规则来中止未完成分段上传的桶。然后，您可以在 S3 控制台中导航到这些桶，并添加生命周期规则以删除已丢弃的未完成分段上传。

步骤 3：添加生命周期规则以在 7 天后删除未完成分段上传

要自动管理未完成的分段上传，您可以使用 S3 控制台创建生命周期配置，以使桶中的未完成分段上传字节在指定天数后过期。有关更多信息，请参阅配置存储桶生命周期配置以删除未完成的分段上传。

减少保留的非当前版本的数量

启用后，S3 版本控制会保留同一对象的多个不同版本，当对象被意外删除或覆盖时，可以使用这些版本快速恢复数据。如果您在未配置生命周期规则以使非当前版本转换或过期的情况下启用了 S3 版本控制，则会累积大量以前的非当前版本，这可能会对存储成本产生影响。有关更多信息，请参阅在 S3 存储桶中使用版本控制。

步骤 1：识别具有最多非当前对象版本的桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要查看的控制面板。
4. 在 Snapshot for date（日期的快照）部分中的 Metric categories（指标类别）下，选择 Cost optimization（成本优化）。

   在 Snapshot for date（日期的快照）部分将更新，以显示 Cost optimization（成本优化）指标，其中包括 % noncurrent version bytes（非当前版本字节百分比）的指标。% noncurrent version

bytes（非当前版本字节百分比）指标表示控制面板范围中在所选日期内归属于非当前版本的总存储字节的比例。

> **Note**
>
> 如果 % noncurrent version bytes（非当前版本字节百分比）大于账户级存储的 10%，则可能表明您所存储的对象版本太多。

5. 要确定积累了大量非当前版本的特定桶，请执行以下操作：

   a. 向下滚动到 Top N overview for date（日期的前 N 个概览）部分。对于 Top N（前 N 个），输入您想要查看其数据的桶数。

   b. 对于 Metric（指标），选择 % noncurrent version bytes（非当前版本字节百分比）。

      在 Top number buckets（前 number 个桶）下，您可以看到 % noncurrent version bytes（非当前版本字节百分比）最高的桶（针对您指定的数字）。Top N overview for date（日期的前 N 个概览）部分还显示与前一天或前一周相比的百分比变化以及用于可视化趋势的火花线。该趋势对于免费指标为 14 天趋势，对于高级指标和建议为 30 天趋势。

      > **Note**
      >
      > 使用 S3 Storage Lens 存储统计管理工具高级指标和建议，指标可用于 15 个月的查询。有关更多信息，请参阅指标选择。

   c. 要查看非当前对象版本的更详细的桶级指标，请滚动到页面顶部，然后选择 Bucket（桶）选项卡。

      在 S3 Storage Lens 存储统计管理工具控制面板中的任何图表或可视化效果中，您都可以使用 Account（账户）、AWS 区域、Storage class（存储类）或 Bucket（桶）选项卡深入到更深层次的聚合。有关示例，请参阅 发现冷的 Amazon S3 桶。

   d. 在 Buckets（桶）部分中，对于 Metric categories（指标类别），选择 Cost optimization（成本优化）。然后，清除 Summary（摘要）。

      现在，您可以看到 % noncurrent version bytes（非当前版本字节百分比）指标，以及与非当前版本相关的其他指标。

步骤 2：确定缺少用于管理非当前版本的转换和过期生命周期规则的桶

先决条件

要在 S3 Storage Lens 存储统计管理工具控制面板中查看 Noncurrent version transition lifecycle rule count（非当前版本转换生命周期规则计数）和 Noncurrent version expiration lifecycle rule count（非当前版本过期生命周期规则计数）指标，必须启用 S3 Storage Lens 存储统计管理工具 Advanced metrics and recommendations（高级指标和建议），然后选择 Advanced cost optimization metrics（高级成本优化指标）。有关更多信息，请参阅创建和更新 Amazon S3 Storage Lens 存储统计管理工具控制面板。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要查看的控制面板。
4. 在 Storage Lens 存储统计管理工具控制面板中，选择 Bucket（桶）选项卡。
5. 向下滚动到 Bucket（桶）部分。对于 Metrics categories（指标类别），选择 Cost optimization（成本优化）。然后清除 Summary（摘要）。

   Buckets（桶）列表将更新，以显示所显示的桶的所有可用 Cost optimization（成本优化）指标。
6. 要筛选 Buckets（桶）列表以仅显示特定的成本优化指标，请选择首选项图标（ ）。
7. 清除所有成本优化指标的切换开关，直到只有以下指标保持选中状态：
   - % noncurrent version bytes（非当前版本字节百分比）
   - Noncurrent version transition lifecycle rule count（非当前版本转换生命周期规则计数）
   - Noncurrent version expiration lifecycle rule count（非当前版本过期生命周期规则计数）
8. （可选）在 Page size（页面大小）下，选择要在列表中显示的桶数。
9. 选择 Confirm（确认）。

   Buckets（桶）列表将更新，以显示非当前版本字节和非当前版本生命周期规则计数的指标。您可以使用这些数据来识别非当前版本字节百分比高但缺少转换和过期生命周期规则的桶。然后，您可以在 S3 控制台中导航到这些桶，并将生命周期规则添加到这些桶。

步骤 3：添加转换非当前对象版本或使其过期的生命周期规则

确定哪些桶需要进一步调查后，您可以导航到 S3 控制台中的桶，并添加生命周期规则以使非当前版本在指定天数后过期。或者，为了降低成本，同时仍然保留非当前版本，您可以配置生命周期规则将非当前版本转换到 Amazon S3 Glacier 存储类之一。有关更多信息，请参阅示例 6：为启用了版本控制的存储桶指定生命周期规则。

识别没有生命周期规则的桶并查看生命周期规则计数

S3 Storage Lens 存储统计管理工具提供 S3 生命周期规则计数指标，您可以使用这些指标来识别缺少生命周期规则的桶。要查找没有生命周期规则的桶，您可以使用 Total buckets without lifecycle rules（无生命周期规则的桶总数）指标。没有 S3 生命周期配置的桶可能有您不再需要的存储，或者可以迁移到成本较低的存储类。您还可以使用生命周期规则计数指标来识别缺少特定类型的生命周期规则（例如过期或转换规则）的桶。

先决条件

要在 S3 Storage Lens 存储统计管理工具控制面板中查看生命规则计数指标和 Total buckets without lifecycle rules（无生命周期规则的桶总数）指标，您必须启用 S3 Storage Lens 存储统计管理工具 Advanced metrics and recommendations（高级指标和建议），然后选择 Advanced cost optimization metrics（高级成本优化指标）。有关更多信息，请参阅创建和更新 Amazon S3 Storage Lens 存储统计管理工具控制面板。

步骤 1：识别没有生命周期规则的桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要查看的控制面板。
4. 要识别没有生命周期规则的特定桶，请向下滚动到 Top N overview for date（日期的前 N 个概览）部分。

   原定设置情况下，Top N overview for date（日期的前 N 个概览）部分显示前 3 个桶的指标。在 Top N（前 N 个）字段中，您可以增加桶的数量。Top N overview for date（日期的前 N 个概览）部分还显示与前一天或前一周相比的百分比变化以及用于可视化趋势的火花线。该趋势对于免费指标为 14 天趋势，对于高级指标和建议为 30 天趋势。

> **Note**
>
> 使用 S3 Storage Lens 存储统计管理工具高级指标和建议，指标可用于 15 个月的查询。有关更多信息，请参阅指标选择。

5. 对于 Metric（指标），从 Cost optimization（成本优化）类别中选择 Total buckets without lifecycle rules（没有生命周期规则的桶总数）。
6. 查看 Total buckets without lifecycle rules（没有生命周期规则的桶总数）的以下数据：

   - Top number accounts（前 number 个账户）- 查看哪些账户拥有最多的没有生命周期规则的桶。
   - Top number Regions（前 number 个区域）- 按区域查看没有生命周期规则的桶的明细。
   - Top number buckets（前 number 个桶）- 查看哪些桶没有生命周期规则。

   在 S3 Storage Lens 存储统计管理工具控制面板中的任何图表或可视化效果中，您都可以使用 Account（账户）、AWS 区域、Storage class（存储类）或 Bucket（桶）选项卡深入到更深层次的聚合。有关示例，请参阅 发现冷的 Amazon S3 桶。

   在识别哪些桶没有生命周期规则后，您还可以查看桶的特定生命周期规则计数。

步骤 2：查看桶的生命周期规则计数

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要查看的控制面板。
4. 在 S3 Storage Lens 存储统计管理工具控制面板中，选择 Bucket（桶）选项卡。
5. 向下滚动到 Bucket（桶）部分。在 Metrics categories（指标类别）下，选择 Cost optimization（成本优化）。然后清除 Summary（摘要）。

   Buckets（桶）列表将更新，以显示所显示的桶的所有可用 Cost optimization（成本优化）指标。
6. 要筛选 Buckets（桶）列表以仅显示特定的成本优化指标，请选择首选项图标（ ）。
7. 清除所有成本优化指标的切换开关，直到只有以下指标保持选中状态：

- Transition lifecycle rule count（转换生命周期规则计数）
- Expiration lifecycle rule count（过期生命周期规则计数）
- Noncurrent version transition lifecycle rule count（非当前版本转换生命周期规则计数）
- Noncurrent version expiration lifecycle rule count（非当前版本过期生命周期规则计数）
- Abort incomplete multipart upload lifecycle rule count（中止未完成分段上传生命周期规则计数）
- Total lifecycle rule count（生命周期规则总计数）

8. （可选）在 Page size（页面大小）下，选择要在列表中显示的桶数。
9. 选择 Confirm（确认）。

   Buckets（桶）列表会更新，以显示您的桶的生命周期规则计数指标。您可以使用这些数据来识别没有生命周期规则的桶或缺少特定类型的生命周期规则（例如过期或转换规则）的桶。然后，您可以在 S3 控制台中导航到这些桶，并将生命周期规则添加到这些桶。

步骤 3：添加生命周期规则

识别没有生命周期规则的桶后，您可以添加生命周期规则。有关更多信息，请参阅 在存储桶上设置生命周期配置 和 S3 生命周期配置的示例：

使用 S3 Storage Lens 存储统计管理工具保护您的数据

您可以使用 Amazon S3 Storage Lens 存储统计管理工具数据保护指标来识别尚未应用数据保护最佳实践的桶。您可以使用这些指标来采取措施并应用符合最佳实践的标准设置，以保护您的账户或组织的桶中的数据。例如，您可以使用数据保护指标来识别不使用 AWS Key Management Service（AWS KMS）密钥（SSE-KMS）进行原定设置加密的桶或使用 AWS 签名版本 2（SigV2）的请求。

以下使用案例提供使用 S3 Storage Lens 存储统计管理工具控制面板识别异常值并跨 S3 桶应用数据保护最佳实践的策略。

主题

- 识别不使用具有 AWS KMS 的服务器端加密（SSE-KMS）进行原定设置加密的桶
- 识别已启用 S3 版本控制的桶
- 识别使用 AWS 签名版本 2（SigV2）的请求
- 计算每个桶的复制规则总计数
- 确定对象锁定字节的百分比

识别不使用具有 AWS KMS 的服务器端加密（SSE-KMS）进行原定设置加密的桶

借助 Amazon S3 原定设置加密，您可以设置 S3 桶的原定设置加密行为。有关更多信息，请参阅the section called “设置默认存储桶加密”。

您可以使用 SSE-KMS enabled bucket count（启用 SSE-KMS 的桶计数）和 % SSE-KMS enabled buckets（启用 SSE-KMS 的桶的百分比）指标来识别使用具有 AWS KMS 密钥的服务器端加密（SSE-KMS）进行原定设置加密的桶。S3 Storage Lens 存储统计管理工具还提供关于未加密字节、未加密对象、加密字节和加密对象的指标。要获得指标的完整列表，请参阅 Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

您可以在常规加密指标的上下文中分析 SSE-KMS 加密指标，以识别不使用 SSE-KMS 的桶。如果您想对账户或组织中的所有桶使用 SSE-KMS，则可以将这些桶的原定设置加密设置更新为使用 SSE-KMS。除了 SSE-KMS 之外，您还可以将服务器端加密与 Amazon S3 托管式密钥（SSE-S3）或与客户托管式密钥（SSE-C）结合使用。有关更多信息，请参阅利用加密来保护数据。

步骤 1：确定哪些桶使用 SSE-KMS 进行原定设置加密

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。
4. 在 Trends and distributions（趋势和分布）部分中，为主要指标选择 % SSE-KMS enabled bucket count（启用 SSE-KMS 的桶计数百分比），为辅助指标选择 % encrypted bytes（加密字节百分比）。

   Trend for date（日期的趋势）图更新，以显示 SSE-KMS 和加密字节的趋势。
5. 要查看 SSE-KMS 的更精细桶级见解，请执行以下操作：

   a. 在图上选择一个点。此时将出现一个方框，其中包含可提供更精细见解的选项。

   b. 选择 Buckets（桶）维度。然后选择 Apply（应用）。
6. 在 Distribution by buckets for date（按日期的桶划分的分布）图中，选择 SSE-KMS enabled bucket count（启用 SSE-KMS 的桶计数）指标。
7. 现在，您可以查看哪些桶启用了 SSE-KMS，哪些桶未启用。

步骤 2：更新桶原定设置加密设置

现在，您已经确定了哪些桶在 % encrypted bytes（加密字节百分比）的上下文中使用 SSE-KMS，您可以识别不使用 SSE-KMS 的桶。然后，您可以选择在 S3 控制台中导航到这些桶，并更新其原定设置加密设置以使用 SSE-KMS 或 SSE-S3。有关更多信息，请参阅配置默认加密。

识别已启用 S3 版本控制的桶

启用后，S3 版本控制功能会保留同一对象的多个版本，当对象被意外删除或覆盖时，可以使用这些版本快速恢复数据。您可以使用 Versioning-enabled bucket count（启用版本控制的桶计数）指标来查看哪些桶使用 S3 版本控制。然后，您可以在 S3 控制台中执行操作，从而为其他桶启用 S3 版本控制。

步骤 1：识别已启用 S3 版本控制的桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。
4. 在 Trends and distributions（趋势和分布）部分中，为主要指标选择 Versioning-enabled bucket count（启用版本控制的桶计数），为辅助指标选择 Buckets（桶）。

   Trend for date（日期的趋势）图更新，以显示启用 S3 版本控制的桶的趋势。在趋势线的正下方，您可以看到 Storage class distribution（存储类分布）和 Region distribution（区域分布）子部分。
5. 要查看您在 Trend for date（日期的趋势）图中看到的任何桶的更精细见解，以便进行更深入的分析，请执行以下操作：

   a. 在图上选择一个点。此时将出现一个方框，其中包含可提供更精细见解的选项。

   b. 选择一个维度应用于您的数据以进行更深入的分析：Account（账户）、AWS 区域、Storage class（存储类）或 Bucket（桶）。然后选择 Apply（应用）。
6. 在 Bubble analysis by buckets for date（按日期的桶划分的气泡分析）部分中，选择 Versioning-enabled bucket count（启用版本控制的桶计数）、Buckets（桶）和 Active buckets（活动的桶）指标。

   Bubble analysis by buckets for date（按日期的桶划分的气泡分析）部分将更新，以显示所选指标的数据。您可以使用这些数据在桶总计数的上下文中查看哪些桶启用了 S3 版本控制。在 Bubble analysis by buckets for date（按日期的桶划分的气泡分析）部分中，您可以使用任意三个指标在多个维度上绘制桶，以表示气泡的 X-axis（X 轴）、Y-axis（Y 轴）和 Size（大小）。

步骤 2：启用 S3 版本控制

识别启用了 S3 版本控制的桶后，您可以识别从未启用 S3 版本控制或暂停版本控制的桶。然后，您可以选择在 S3 控制台中为这些桶启用版本控制。有关更多信息，请参阅在存储桶上启用版本控制。

识别使用 AWS 签名版本 2（SigV2）的请求

您可以使用 All unsupported signature requests（所有不支持的签名请求）指标来识别使用 AWS 签名版本 2（SigV2）的请求。这些数据可以帮助您识别使用 SigV2 的特定应用程序。然后，您可以将这些应用程序迁移到 AWS 签名版本 4（SigV4）。

SigV4 是所有新 S3 应用程序的推荐签名方法。SigV4 可提供更高的安全性，并且在所有 AWS 区域中得到支持。有关更多信息，请参阅 Amazon S3 更新 - SigV2 弃用期延长并修改。

先决条件

要在 S3 Storage Lens 存储统计管理工具控制面板中查看 All unsupported signature requests（所有不支持的签名请求），您必须启用 S3 Storage Lens 存储统计管理工具 Advanced metrics and recommendations（高级指标和建议），然后选择 Advanced data protection metrics（高级数据保护指标）。有关更多信息，请参阅创建和更新 Amazon S3 Storage Lens 存储统计管理工具控制面板。

步骤 1：按 AWS 账户、区域和桶检查 SigV2 签名趋势

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。
4. 要识别具有使用 SigV2 的请求的特定桶、账户和区域，请执行以下操作：

    a. 在 Top N overview for date（日期的前 N 个概览）下的 Top N（前 N 个）中，输入您想要查看其数据的桶数。

    b. 对于 Metric（指标），从 Data protection（数据保护）类别中选择 All unsupported signature requests（所有不支持的签名请求）。

    Top N overview for date（日期的前 N 个概览）将更新，以按账户、AWS 区域和桶显示 SigV2 请求的数据。Top N overview for date（日期的前 N 个概览）部分还显示与前一天或前一周相比的百分比变化以及用于可视化趋势的火花线。该趋势对于免费指标为 14 天趋势，对于高级指标和建议为 30 天趋势。

> **Note**
>
> 使用 S3 Storage Lens 存储统计管理工具高级指标和建议，指标可用于 15 个月的查询。有关更多信息，请参阅指标选择。

步骤 2：识别应用程序通过 SigV2 请求访问的桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。
4. 在 Storage Lens 存储统计管理工具控制面板中，选择 Bucket（桶）选项卡。
5. 向下滚动到 Bucket（桶）部分。在 Metrics categories（指标类别）下，选择 Data protection（数据保护）。然后清除 Summary（摘要）。

   Buckets（桶）列表将更新，以显示所显示的桶的所有可用 Data protection（数据保护）指标。
6. 要筛选 Buckets（桶）列表以仅显示特定的数据保护指标，请选择首选项图标（  ）。
7. 清除所有数据保护指标的切换开关，直到只有以下指标保持选中状态：
   - All unsupported signature requests（所有不支持的签名请求）
   - % all unsupported signature requests（所有不支持的签名请求百分比）
8. （可选）在 Page size（页面大小）下，选择要在列表中显示的桶数。
9. 选择 Confirm（确认）。

   Buckets（桶）列表将更新以显示 SigV2 请求的桶级指标。您可以使用这些数据来识别具有 SigV2 请求的特定桶。然后，您可以使用此信息将应用程序迁移到 SigV4。有关更多信息，请参阅《Amazon Simple Storage Service API》参考中的验证请求 (AWS Signature Version 4)。

计算每个桶的复制规则总计数

S3 复制支持跨 Amazon S3 桶自动以异步方式复制对象。为对象复制配置的桶可由相同 AWS 账户 或不同账户拥有。有关更多信息，请参阅复制对象概述。

您可以使用 S3 Storage Lens 存储统计管理工具复制规则计数指标来获取有关配置为进行复制的桶的每个桶的详细信息。此信息包括桶和区域内部以及跨桶和区域的复制规则。

先决条件

要在 S3 Storage Lens 存储统计管理工具控制面板中查看复制规则计数指标，您必须启用 S3 Storage Lens 存储统计管理工具 Advanced metrics and recommendations（高级指标和建议），然后选择 Advanced data protection metrics（高级数据保护指标）。有关更多信息，请参阅创建和更新 Amazon S3 Storage Lens 存储统计管理工具控制面板。

步骤 1：计算每个桶的复制规则总计数

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。
4. 在 Storage Lens 存储统计管理工具控制面板中，选择 Bucket（桶）选项卡。
5. 向下滚动到 Bucket（桶）部分。在 Metrics categories（指标类别）下，选择 Data protection（数据保护）。然后清除 Summary（摘要）。
6. 要筛选 Buckets（桶）列表以仅显示复制规则计数指标，请选择首选项图标（ ）。
7. 清除所有数据保护指标的切换开关，直到只有复制规则计数指标保持选中状态：
   - Same-Region Replication rule count（同区域复制规则计数）
   - Cross-Region Replication rule count（跨区域复制规则计数）
   - Same-account replication rule count（同账户复制规则计数）
   - Cross-account replication rule count（跨账户复制规则计数）
   - Total replication rule count（复制规则总计数）
8. （可选）在 Page size（页面大小）下，选择要在列表中显示的桶数。
9. 选择 Confirm（确认）。

步骤 2：添加复制规则

获得每个桶的复制规则计数后，您可以选择创建其他复制规则。有关更多信息，请参阅配置实时复制的示例。

确定对象锁定字节的百分比

借助 S3 对象锁定，您可以使用一次写入，多次读取（WORM）模式存储对象。您可以使用对象锁定帮助您防止在固定的时间段内或无限期地删除或覆盖对象。您只能在创建桶时启用对象锁定，也可以启用 S3 版本控制。但是，您可以编辑各个对象版本的保留期，也可以对启用了对象锁定的桶应用依法保留。有关更多信息，请参阅使用 S3 对象锁定。

您可以使用 S3 Storage Lens 存储统计管理工具中的对象锁定指标来查看您的账户或组织的 % Object Lock bytes（对象锁定字节百分比）指标。您可以使用这些信息来识别您的账户或组织中未遵循数据保护最佳实践的桶。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。
4. 在 Snapshot（快照）部分的 Metrics categories（指标类别）下，选择 Data protection（数据保护）。

   Snapshot（快照）部分更新以显示数据保护指标，包括 % Object Lock bytes（对象锁定字节百分比）指标。您可以查看您的账户或组织的对象锁定字节的总体百分比。
5. 要查看每个桶的 % Object Lock bytes（对象锁定字节百分比），请向下滚动到 Top N overview（前 N 个概览）部分。

   要获取对象锁定的对象级数据，您还可以使用 Object Lock object count（对象锁定对象计数）和 % Object Lock objects（对象锁定对象百分比）指标。
6. 对于 Metric（指标），从 Data protection（数据保护）类别中选择 % Object Lock bytes（对象锁定字节百分比）。

   原定设置情况下，Top N overview for date（日期的前 N 个概览）部分显示前 3 个桶的指标。在 Top N（前 N 个）字段中，您可以增加桶的数量。Top N overview for date（日期的前 N 个概览）部分还显示与前一天或前一周相比的百分比变化以及用于可视化趋势的火花线。该趋势对于免费指标为 14 天趋势，对于高级指标和建议为 30 天趋势。

> **Note**
>
> 使用 S3 Storage Lens 存储统计管理工具高级指标和建议，指标可用于 15 个月的查询。有关更多信息，请参阅指标选择。

7. 查看 % Object Lock bytes（对象锁定字节百分比）的以下数据：

   - Top number accounts（前 number 个账户）- 查看哪些账户具有最高和最低的 % Object Lock bytes（对象锁定字节百分比）。
   - Top number Regions（前 number 个区域）- 查看按区域划分的 % Object Lock bytes（对象锁定字节百分比）明细。
   - Top number buckets（前 number 个桶）-查看哪些桶具有最高和最低的 % Object Lock bytes（对象锁定字节百分比）。

使用 S3 Storage Lens 存储统计管理工具审计对象所有权设置

Amazon S3 对象所有权是 S3 桶级别的设置，您可以使用它来禁用访问控制列表（ACL）并控制桶中对象的所有权。如果将对象所有权设置为“强制桶拥有者”，则可以禁用访问控制列表（ACL）并获取桶中每个对象的所有权。这种方法简化了对存储在 Amazon S3 中的数据的访问管理。

预设情况下，当另一个 AWS 账户 将对象上载到您的 S3 桶，该账户（对象写入器）拥有该对象，拥有对象的访问权限，并可以授予其他用户通过 ACL 访问该数据元的权限。您可以使用对象所有权来更改此原定设置行为。

Amazon S3 中的大多数现代使用案例不再需要使用 ACL。因此，我们建议您禁用 ACL，除非出现必须单独控制每个对象的访问权限的异常情况。通过将对象所有权设置为“强制桶拥有者”，您可以禁用 ACL 并依赖策略进行访问控制。有关更多信息，请参阅为您的存储桶控制对象所有权和禁用 ACL。。

使用 S3 Storage Lens 存储统计管理工具访问管理指标，您可以识别未禁用 ACL 的桶。识别这些桶后，您可以将 ACL 权限迁移到策略并对这些桶禁用 ACL。

主题

步骤 1：确定对象所有权设置的一般趋势

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。

4. 在 Snapshot for date（日期的快照）部分中的 Metrics categories（指标类别）下，选择 Access management（访问管理）。

   Snapshot for date（日期的快照）部分将更新，以显示 % Object Ownership bucket owner enforced（对象所有权强制桶拥有者的百分比）指标。您可以看到账户或组织中为对象所有权使用“强制桶拥有者”设置以禁用 ACL 的桶的总体百分比。

步骤 2：确定对象所有权设置的桶级趋势

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。
4. 要查看更详细的桶级指标，请选择 Bucket（桶）选项卡。
5. 在 Distribution by buckets for date（按日期的桶划分的分布）部分中，选择 % Object Ownership bucket owner enforced（对象所有权强制桶拥有者的百分比）指标。

   图表将更新以显示 % Object Ownership bucket owner enforced（对象所有权强制桶拥有者的百分比）的每个桶明细。您可以看到哪些桶对于对象所有权使用“强制桶拥有者”设置来禁用 ACL。
6. 要在上下文中查看“强制桶拥有者”设置，请向下滚动到 Buckets（桶）部分。对于 Metrics categories（指标类别），选择 Access management（访问管理）。然后清除 Summary（摘要）。

   Buckets（桶）列表显示所有三种对象所有权设置的数据：强制桶拥有者、首选桶拥有者以及对象写入器。
7. 要筛选 Buckets（桶）列表以仅显示特定对象所有权设置的指标，请选择首选项图标（ ）。
8. 清除您不想看到的指标。
9. （可选）在 Page size（页面大小）下，选择要在列表中显示的桶数。
10. 选择 Confirm（确认）。

步骤 3：将您的对象所有权设置更新为“强制桶拥有者”以禁用 ACL

在确定对于对象所有权使用对象写入器和首选桶拥有者设置后，您可以将 ACL 权限迁移到桶策略。完成 ACL 权限迁移后，您可以将对象所有权设置更新为“强制桶拥有者”以禁用 ACL。有关更多信息，请参阅禁用 ACL 的先决条件。

使用 S3 Storage Lens 存储统计管理工具指标提高性能

如果启用了 S3 Storage Lens 存储统计管理工具高级指标，您可以使用详细的状态代码指标来了解成功或失败的请求的计数。您可以使用此信息来排查访问或性能问题。详细的状态代码指标显示 HTTP 状态代码的计数，例如 403 Forbidden（403 禁止）和 503 Service Unavailable（503 服务不可用）。您可以查看跨 S3 桶、账户和组织的详细状态代码指标的总体趋势。然后，您可以深入了解桶级别的指标，以识别当前正在访问这些桶并导致错误的工作负载。

例如，您可以查看 403 Forbidden error count（403 禁止错误计数）指标，以确定在未应用正确权限的情况下访问桶的工作负载。识别这些工作负载后，您可以在 S3 Storage Lens 存储统计管理工具之外进行深入探索，以对 403 Forbidden（403 禁止）错误进行故障排除。

此示例向您展示如何使用 403 Forbidden error count（403 禁止错误计数）和 % 403 Forbidden errors（403 禁止错误百分比）指标对 403 Forbidden（403 禁止）错误进行趋势分析。您可以使用这些指标来识别在未应用正确权限的情况下访问桶的工作负载。您可以对任何其他 Detailed status code metrics（详细状态代码指标）进行类似的趋势分析。有关更多信息，请参阅Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

先决条件

要在 S3 Storage Lens 存储统计管理工具控制面板中查看 Detailed status code metrics（详细状态代码指标），您必须启用 S3 Storage Lens 存储统计管理工具 Advanced metrics and recommendations（高级指标和建议），然后选择 Detailed status code metrics（详细状态代码指标）。有关更多信息，请参阅创建和更新 Amazon S3 Storage Lens 存储统计管理工具控制面板。

主题

步骤 1：对单个 HTTP 状态代码进行趋势分析

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。
4. 在 Trends and distributions（趋势和分布）部分中，对于 Primary metric（主要指标），从 Detailed status codes（详细状态代码）类别中选择 403 Forbidden error count（403 禁止错误计数）。对于 Secondary metric（辅助指标），选择 % 403 Forbidden errors（403 禁止错误百分比）。
5. 向下滚动到 Top N overview for date（日期的前 N 个概览）部分。对于 Metrics（指标），从 Detailed status codes（详细状态代码）类别中选择 403 Forbidden error count（403 禁止错误计数）或 % 403 Forbidden errors（403 禁止错误百分比）。

   Top N overview for date（日期的前 N 个概览）部分将更新，以显示按账户、AWS 区域 和桶显示排名靠前的 403 禁止错误计数。

步骤 2：按桶分析错误计数

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在 Dashboards（控制面板）列表中，选择您要查看的控制面板的名称。
4. 在 Storage Lens 存储统计管理工具控制面板中，选择 Bucket（桶）选项卡。
5. 向下滚动到 Bucket（桶）部分。对于 Metrics categories（指标类别），选择 Detailed status code（详细状态代码）指标。然后清除 Summary（摘要）。

   Buckets（桶）列表将更新，以显示所有可用的详细状态代码指标。您可以使用这些信息来查看哪些桶具有很大比例的特定 HTTP 状态代码，以及哪些状态代码在桶中是常见的。
6. 要筛选 Buckets（桶）列表以仅显示特定的详细状态代码指标，请选择首选项图标（  ）。
7. 清除您不想在 Buckets（桶）列表中看到的任何详细状态代码指标的切换开关。
8. （可选）在 Page size（页面大小）下，选择要在列表中显示的桶数。
9. 选择 Confirm（确认）。

Buckets（桶）列表显示您指定的桶数量的错误计数指标。您可以使用这些信息来识别出现许多错误的特定桶，并按桶排除错误。

步骤 3：排除错误

在识别出具有高比例的特定 HTTP 状态代码的桶后，您可以排除这些错误。有关更多信息，请参阅下列内容：

- 当我尝试在 Amazon S3 中上载文件时，为什么会出现 403 Forbidden（403 禁止）错误？
- 当我尝试在 Amazon S3 中修改桶策略时，为什么会出现 403 Forbidden（403 禁止）错误？
- 如何排除所有资源都来自相同 AWS 账户的 Amazon S3 桶中的 403 Forbidden（403 禁止）错误？
- 如何排除来自 Amazon S3 的 HTTP 500 或 503 错误？

Amazon S3 Storage Lens 存储统计管理工具指标词汇表

Amazon S3 Storage Lens 存储统计管理工具指标词汇表提供了 S3 Storage Lens 存储统计管理工具的免费和高级指标的完整列表。

S3 Storage Lens 存储统计管理工具为所有控制面板和配置提供免费指标以及升级到高级指标的选项。

- 免费指标包含与存储使用情况相关的指标，例如账户中的桶数量和对象数量。免费指标还包括基于应用场景的指标，例如成本优化和数据保护指标。每天都会收集所有免费指标，数据在最多 14 天内可供查询。
- 高级指标和建议包括免费指标中的所有指标以及其它指标，例如高级数据保护和成本优化指标。高级指标还包括其它指标类别，例如活动指标和详细状态代码指标。高级指标数据可供查询 15 个月。

  如果您使用 S3 Storage Lens 存储统计管理工具和高级指标和建议，则需要支付额外费用。有关更多信息，请参阅 Amazon S3 定价。有关高级指标和建议功能的更多信息，请参阅指标选择。

> **Note**
>
> 对于 Storage Lens 组，只有免费存储指标可用。高级指标在 Storage Lens 组级别不可用。

指标名称

下表中的 Metric name（指标名称）列提供了 S3 控制台中每个 S3 Storage Lens 存储统计管理工具的名称。CloudWatch and export（CloudWatch 和导出）列提供了 Amazon CloudWatch 中每个指标的名称以及您可以在 S3 Storage Lens 存储统计管理工具控制面板中配置的指标导出文件。

派生指标公式

派生指标不适用于指标导出和 CloudWatch 发布选项。但是，您可以使用 Derived metrics formula（派生指标公式）列中所示的指标公式来计算它们。

解释 Amazon S3 Storage Lens 存储统计管理工具的指标单位倍数的前缀符号（K、M、G 等）

S3 Storage Lens 存储统计管理工具指标单位倍数用前缀符号编写。这些前缀符号与由国际计量局（BIPM）标准化的国际单位制（SI）符号相符。这些符号还用于统一计量单位代码（UCUM)中。有关更多信息，请参阅 SI 前缀符号列表。

> **Note**
>
> - S3 存储字节的计量单位是二进制千兆字节（GB），其中 1GB 是 2^{30} 字节，1TB 是 2^{40} 字节，1PB 是 2^{50} 字节。根据国际电工委员会（IEC）的定义，该计量单位也称为吉比字节（GiB）。
> - 在对象根据其生命周期配置的生存期结束后，Amazon S3 会将该对象加入移除队列并异步移除它。因此，过期日期和 Amazon S3 删除对象的日期之间可能会有一段延迟。S3 Storage Lens 存储统计管理工具不包含已过期但尚未移除的对象的指标。有关 S3 生命周期中的过期操作的更多信息，请参阅即将过期的对象。

S3 Storage Lens 存储统计管理工具指标词汇表

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
总存储空间	StorageBytes	总存储空间，包括未完成的分段上传、对象元数据和删除标记	免费	Sum	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
对象计数	ObjectCount	对象总数	免费	Sum	否	-
平均对象大小	-	对象平均大小	免费	Sum	Y	sum(StorageBytes)/sum(ObjectCount)
处于活动状态的桶数	-	存储大于 0 字节的存储桶总数	免费	Sum	Y	-
桶	-	桶的总数	免费	Sum	Y	-
账户	-	其存储在范围内的账户数量	免费	Sum	Y	-
当前版本字节数	CurrentVersionStorageBytes	属于对象的当前版本的字节数	免费	成本优化	否	-
当前版本字节百分比	-	范围内属于对象的当前版本的字节百分比	免费	成本优化	Y	sum(CurrentVersionStorageBytes)/sum(StorageBytes)

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
当前版本对象计数	CurrentVersionObjectCount	当前版本对象的计数	免费	成本优化	否	-
当前版本对象百分比	-	范围内属当前版本的对象的百分比	免费	成本优化	Y	sum(CurrentVersionObjectCount)/sum(ObjectCount)
非当前版本字节数	NonCurrentVersionStorageBytes	非当前版本的字节数	免费	成本优化	否	-
非当前版本字节百分比	-	范围内非当前版本字节百分比	免费	成本优化	Y	sum(NonCurrentVersionStorageBytes)/sum(StorageBytes)
非当前版本对象计数	NonCurrentVersionObjectCount	非当前版本对象的计数	免费	成本优化	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
非当前版本对象百分比	-	范围内非当前版本对象百分比	免费	成本优化	Y	sum(NonCurrentVersionObjectCount)/sum(ObjectCount)
删除标记字节	DeleteMarkerStorageBytes	范围内作为删除标记的字节数	免费	成本优化	否	-
删除标记字节百分比	-	范围内作为删除标记的字节百分比	免费	成本优化	Y	sum(DeleteMarkerStorageBytes)/sum(StorageBytes)
删除标记对象计数	DeleteMarkerObjectCount	带有删除标记的对象总数	免费	成本优化	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
删除标记对象百分比	-	范围内带有删除标记的对象百分比	免费	成本优化	Y	sum(DeleteMarkerObjectCount)/sum(ObjectCount)
未完成分段上传字节	IncompleteMultipartUploadStorageBytes	范围内未完成分段上传的总字节数	免费	成本优化	否	-
未完成分段上传字节百分比	-	范围内由于未完成分段上传导致的字节百分比	免费	成本优化	Y	sum(IncompleteMultipartUploadStorageBytes)/sum(StorageBytes)
未完成分段上传对象计数	IncompleteMultipartUploadObjectCount	范围内未完成分段上传的对象数量	免费	成本优化	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
未完成分段上传对象百分比	-	范围内未完成分段上传的对象百分比	免费	成本优化	Y	sum(IncompleteMultipartUploadObjectCount)/sum(ObjectCount)
超过 7 天的未完成分段上传存储字节	IncompleteMPUStorageBytesOlderThan7Days	范围内超过 7 天的未完成分段上传的总字节数	免费	成本优化	否	-
超过 7 天的未完成分段上传存储字节百分比	-	超过 7 天的未完成分段上传字节的百分比	免费	成本优化	Y	sum(IncompleteMPUStorageBytesOlderThan7Days)/sum(StorageBytes)
超过 7 天的未完成分段上传对象计数	IncompleteMPUObjectCountOlderThan7Days	超过 7 天的未完成分段上传的对象数量	免费	成本优化	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
超过 7 天的未完成分段上传对象计数百分比	-	超过 7 天的未完成分段上传对象的百分比	免费	成本优化	Y	sum(IncompleteMPUObjectCountOlderThan7Days)/sum(ObjectCount)
转换生命周期规则计数	TransitionLifecycleRuleCount	将对象转换为另一个存储类的生命周期规则计数	高级	成本优化	否	-
每个桶的平均转换生命周期规则数	-	将对象转换为另一个存储类的生命周期规则的平均数量	高级	成本优化	Y	sum(TransitionLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
过期生命周期规则计数	ExpirationLifecycleRuleCount	使对象过期的生命周期规则计数	高级	成本优化	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
每个桶的平均过期生命周期规则数	-	使对象过期的生命周期规则的平均数量	高级	成本优化	Y	sum(ExpirationLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
非当前版本转换生命周期规则计数	NoncurrentVersionTransitionLifecycleRuleCount	将非当前对象版本转换为其他存储类的生命周期规则计数	高级	成本优化	否	
每个桶的平均非当前版本转换生命周期规则数	-	将非当前对象版本转换到另一个存储类的生命周期规则的平均数量	高级	成本优化	Y	sum(NoncurrentVersionTransitionLifecycleRuleCount)/sum(DistinctNumberOfBuckets)

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
非当前版本过期生命周期规则计数	NoncurrentVersionExpirationLifecycleRuleCount	使非当前对象版本过期的生命周期规则计数	高级	成本优化	否	-
每个桶的平均非当前版本过期生命周期规则数	-	使非当前对象版本过期的生命周期规则的平均数量	高级	成本优化	Y	sum(NoncurrentVersionExpirationLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
中止未完成的分段上传生命周期规则计数	AbortIncompleteMPULifecycleRuleCount	删除未完成分段上传的生命周期规则计数	高级	成本优化	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
每个桶的中止未完成分段上传生命周期规则的平均数量	-	删除未完成分段上传的生命周期规则的平均数量	高级	成本优化	Y	sum(AbortIncompleteMPULifecycleRuleCount)/sum(DistinctNumberOfBuckets)
过期对象删除标记生命周期规则计数	ExpiredObjectDeleteMarkerLifecycleRuleCount	移除过期对象删除标记的生命周期计数	高级	成本优化	否	-
每个桶的过期对象删除标记生命周期规则的平均数量	-	移除过期对象删除标记的生命周期的平均数量	高级	成本优化	Y	sum(ExpiredObjectDeleteMarkerLifecycleRuleCount)/sum(DistinctNumberOfBuckets)

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
生命周期规则总计数	TotalLifecycleRuleCount	生命周期规则的总计数	高级	成本优化	否	-
每个桶的生命周期规则平均计数	-	生命周期规则平均数量	高级	成本优化	Y	sum(TotalLifecycleRuleCount)/sum(DistinctNumberOfBuckets)
加密字节数	EncryptedStorageBytes	加密字节总数	免费	数据保护	否	-
加密字节百分比	-	已加密总字节数的百分比	免费	数据保护	Y	sum(EncryptedObjectCount)/sum(StorageBytes)
加密对象计数	EncryptedObjectCount	加密对象的总计数	免费	数据保护	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
加密对象百分比	-	加密对象的百分比	免费	数据保护	Y	sum(EncryptedStorageBytes)/sum(ObjectCount)
未加密字节数	UnencryptedStorageBytes	未加密的字节数	免费	数据保护	Y	sum(StorageBytes) - sum(EncryptedStorageBytes)
未加密字节百分比	-	未加密字节的百分比	免费	数据保护	Y	sum(UnencryptedStorageBytes)/sum(StorageBytes)
未加密对象计数	UnencryptedObjectCount	未加密对象的总计数	免费	数据保护	Y	sum(ObjectCount) - sum(EncryptedObjectCount)

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
未加密对象百分比	-	未加密对象的百分比	免费	数据保护	Y	sum(UnencryptedStorageBytes)/ sum(ObjectCount)
复制的存储字节源	ReplicatedStorageBytesSource	从源桶复制的总字节数	免费	数据保护	否	-
复制字节源百分比	-	从源桶复制的总字节数的百分比	免费	数据保护	Y	sum(ReplicatedStorageBytesSource)/ sum(StorageBytes)
复制对象计数源	ReplicatedObjectCountSource	从源桶中复制的对象的计数	免费	数据保护	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
复制对象源百分比	-	从源桶复制的总对象数的百分比	免费	数据保护	Y	sum(ReplicatedStorageObjectCount)/sum(ObjectCount)
复制存储字节目标	ReplicatedStorageBytes	复制到目标桶的总字节数	免费	数据保护	Y	-
复制字节目标百分比	-	复制到目标桶的总字节数的百分比	免费	数据保护	Y	sum(ReplicatedStorageBytes)/sum(StorageBytes)
复制对象计数目标	ReplicatedObjectCount	复制到目标桶的对象计数	免费	数据保护	Y	-
复制对象目标百分比	-	复制到目标桶的总对象数的百分比	免费	数据保护	Y	sum(ReplicatedObjectCount)/sum(ObjectCount)

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
对象锁定字节	ObjectLockEnabledStorageBytes	启用对象锁定的存储字节总计数	免费	数据保护	Y	sum(UnencryptedStorageBytes)/ sum(ObjectLockEnabledStorageCount) - sum(ObjectLockEnabledStorageBytes)
对象锁定字节百分比	-	启用对象锁定的存储字节的百分比	免费	数据保护	Y	sum(ObjectLockEnabledStorageBytes)/ sum(StorageBytes)
对象锁定对象计数	ObjectLockEnabledObjectCount	对象锁定对象的总计数	免费	数据保护	Y	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
对象锁定对象百分比	-	启用了对象锁定的总对象百分比	免费	数据保护	Y	sum(ObjectLockEnabledObjectCount)/sum(ObjectCount)
启用版本控制的桶计数	VersioningEnabledBucketCount	启用 S3 版本控制的桶的计数	免费	数据保护	否	-
启用版本控制的桶百分比	-	启用 S3 版本控制的桶的百分比	免费	数据保护	Y	sum(VersioningEnabledBucketCount)/sum(DistinctNumberOfBuckets)
启用 MFA 删除的桶计数	MFADeleteEnabledBucketCount	启用 MFA（多重验证）删除的桶计数	免费	数据保护	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
启用 MFA 删除的桶百分比	-	启用 MFA（多重验证）删除的桶百分比	免费	数据保护	Y	sum(MFADeleteEnabledBucketCount)/ sum (Distinct NumberOfBuckets)
启用 SSE-KMS 的桶计数	SSEKMSEnabledBucketCount	使用具有 AWS Key Management Service 密钥的服务器端加密（SSE-KMS）进行原定设置桶加密的桶计数	免费	数据保护	否	-
启用 SSE-KMS 的桶百分比	-	使用 SSE-KMS 进行原定设置桶加密的桶百分比	免费	数据保护	Y	sum(SSEKMSEnabledBucketCount)/ sum(DistinctNumberOfBuckets)
所有不支持的签名请求	AllUnsupportedSignatureRequests	使用不支持的 AWS 签名版本的请求总数	高级	数据保护	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
所有不支持的签名请求百分比	-	使用不支持的 AWS 签名版本的请求的百分比	高级	数据保护	Y	sum(AllUnsupportedSignatureRequests) / sum(AllRequests)
所有不支持的 TLS 请求	AllUnsupportedTLSRequests	使用不支持的传输层安全性协议（TLS）版本的请求数量	高级	数据保护	否	-
所有不支持的 TLS 请求百分比	-	使用不支持的 TLS 版本的请求的百分比	高级	数据保护	Y	sum(AllUnsupportedTLSRequests)/ sum(AllRequests)
所有 SSE-KMS 请求	AllSSEKMSRequests	指定 SSE-KMS 的请求总数	高级	数据保护	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
所有 SSE-KMS 请求的百分比	-	指定 SSE-KMS 的请求百分比	高级	数据保护	Y	sum(AllSSEKMSRequests)/sum(AllRequests)
同区域复制规则计数	SameRegionReplicationRuleCount	同区域复制（SRR）的复制规则计数	高级	数据保护	否	-
每个桶的平均同区域复制规则数	-	SRR 的复制规则的平均数量	高级	数据保护	Y	sum(SameRegionReplicationRuleCount)/sum(DistinctNumberOfBuckets)
跨区域复制规则计数	CrossRegionReplicationRuleCount	跨区域复制（CRR）的复制规则计数	高级	数据保护	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
每个桶的平均跨区域复制规则数	-	CRR 的复制规则的平均数量	高级	数据保护	Y	sum(CrossRegionReplicationRuleCount) / sum(DistinctNumberOfBuckets)
同账户复制规则计数	SameAccountReplicationRuleCount	同一账户内用于复制的复制规则计数	高级	数据保护	否	-
每个桶的平均同账户复制规则数	-	同一账户内用于复制的复制规则平均数量	高级	数据保护	Y	sum(SameAccountReplicationRuleCount) / sum(DistinctNumberOfBuckets)
跨账户复制规则计数	CrossAccountReplicationRuleCount	跨账户复制的复制规则计数	高级	数据保护	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
每个桶的跨账户复制规则平均数量	-	跨账户复制的复制规则的平均数量	高级	数据保护	Y	sum(CrossAccountReplicationRuleCount)/sum(DistinctNumberOfBuckets)
目标复制规则计数无效	InvalidDestinationReplicationRuleCount	具有无效复制目标的复制规则计数	高级	数据保护	否	-
每个桶的无效目标复制规则的平均数量	-	具有无效复制目标的复制规则的平均数量	高级	数据保护	Y	sum(InvalidReplicationRuleCount)/sum(DistinctNumberOfBuckets)
复制规则总计数	-	复制规则总计数	高级	数据保护	Y	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
每个桶的复制规则平均计数	-	复制规则总计数的平均值	高级	数据保护	Y	sum(all replication rule count metrics)/ sum(DistinctNumberOfBuckets)
对象所有权强制桶所有者计数	ObjectOwnershipBucketOwnerEnforcedBucketCount	通过为对象所有权使用“强制桶所有者”设置禁用了访问控制列表（ACL）的桶总计数	免费	访问管理	否	-
对象所有权“强制桶所有者”的桶的百分比	-	通过为对象所有权使用“强制桶所有者”设置禁用了 ACL 的桶的百分比	免费	访问管理	Y	sum(ObjectOwnershipBucketOwnerEnforcedBucketCount)/ sum (DistinctNumberOfBuckets)

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
对象所有权“首选桶所有者”的桶的计数	ObjectOwnershipBucketOwnerPreferredBucketCount	为对象所有权使用“首选桶所有者”设置的桶的总计数	免费	访问管理	否	-
对象所有权“首选桶所有者”的桶的百分比	-	为对象所有权使用“首选桶所有者”设置的桶的百分比	免费	访问管理	Y	sum(ObjectOwnershipBucketOwnerPreferredBucketCount)/sum(DistinctNumberOfBuckets)
对象所有权“对象写入器”桶计数	ObjectOwnershipObjectWriterBucketCount	为对象所有权使用“对象写入器”设置的桶的总计数	免费	访问管理	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
对象所有权“对象写入器”桶的百分比	-	为对象所有权使用“对象写入器”设置的桶的百分比	免费	访问管理	Y	sum(ObjectOwnershipObjectWriterBucketCount)/sum(DistinctNumberOfBuckets)
启用 Transfer Acceleration 的桶计数	TransferAccelerationEnabledBucketCount	启用 Transfer Acceleration 的桶的总计数	免费	性能	否	-
启用 Transfer Acceleration 的桶的百分比	-	已启用 Transfer Acceleration 的桶的百分比	免费	性能	Y	sum(TransferAccelerationEnabledBucketCount)/sum(DistinctNumberOfBuckets)
启用事件通知的桶计数	EventNotificationEnabledBucketCount	启用了事件通知的桶的总计数	免费	事件	否	

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
启用事件通知的桶的百分比	-	启用了事件通知的桶的百分比	免费	事件	Y	sum(EventNotificationEnabledBucketCount)/sum(DistinctNumberOfBuckets)
所有请求	AllRequests	发出的 请求总数	高级	活动	否	-
Get 请求	GetRequests	发出的 GET 请求总数	高级	活动	否	-
Put 请求	PutRequests	发出的 PUT 请求总数	高级	活动	否	-
Head 请求	HeadRequests	发出的 HEAD 请求总数	高级	活动	否	-
Delete 请求	DeleteRequests	发出的 DELETE 请求总数	高级	活动	否	-
List 请求	ListRequests	发出的 LIST 请求总数	高级	活动	否	-
Post 请求	PostRequests	发出的 POST 请求总数	高级	活动	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
Select 请求	SelectRequests	S3 Select 请求的总数	高级	活动	否	-
Select 扫描的字节	SelectScannedBytes	扫描的 S3 Select 字节数	高级	活动	否	-
Select 返回的字节	SelectReturnedBytes	返回的 S3 Select 字节数	高级	活动	否	-
已下载字节	BytesDownloaded	下载的字节数	高级	活动	否	-
检索率百分比	-	下载的字节的百分比	高级	活动	Y	sum(BytesDownloaded)/sum(StorageBytes)
已上传字节	BytesUploaded	上传的字节数	高级	活动	否	-
摄取率百分比	-	上载的字节的百分比	高级	活动	Y	sum(BytesUploaded)/sum(StorageBytes)
4xx 错误	4xxErrors	HTTP 4xx 状态代码的总数	高级	活动	否	-

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
5xx 错误	5xxErrors	HTTP 5xx 状态代码的总数	高级	活动	否	-
总错误数	-	所有 4xx 和 5xx 错误的总和	高级	活动	Y	sum(4xxErrors) + sum(5xxErrors)
错误率百分比	-	4xx 和 5xx 错误总数占总请求数的百分比	高级	活动	Y	sum(TotalErrors)/ sum(TotalRequests)
200 OK 状态计数	200OKStatusCount	200 OK 状态代码的总计数	高级	详细状态代码	否	-
% 200 OK 状态百分比	-	200 OK 状态代码总数占总请求数的百分比	高级	详细状态代码	Y	sum(200OKStatusCount)/ sum(AllRequests)

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
206 部分内容状态计数	206PartialContentStatusCount	206 部分内容状态代码的总计数	高级	详细状态代码	否	-
206 部分内容状态百分比	-	206 部分内容状态代码总数占总请求数的百分比	高级	详细状态代码	Y	sum(206PartialContentStatusCount)/sum(AllRequests)
400 错误请求错误计数	400BadRequestErrorCount	400 错误请求状态代码的总计数	高级	详细状态代码	否	-
400 错误请求错误百分比	-	400 错误请求状态代码总数占总请求数的百分比	高级	详细状态代码	Y	sum(400BadRequestErrorCount)/sum(AllRequests)

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
403 禁止错误计数	403ForbiddenErrorCount	403 禁止状态代码的总计数	高级	详细状态代码	否	-
403 禁止错误百分比	-	403 禁止状态代码总数占总请求数的百分比	高级	详细状态代码	Y	sum(403ForbiddenErrorCount) / sum(AllRequests)
404 找不到错误计数	404NotFoundErrorCount	404 找不到状态代码的总计数	高级	详细状态代码	否	-
404 找不到错误百分比	-	404 找不到状态代码总数占总请求数的百分比	高级	详细状态代码	Y	sum(404NotFoundErrorCount)/ sum(AllRequests)

指标名称	CloudWatch 和导出	描述	级别[1]	类别[2]	派生	派生指标公式
500 内部服务器错误计数	500InternalServerErrorCount	500 内部服务器错误状态代码总计数	高级	详细状态代码	否	-
500 内部服务器错误百分比	-	500 内部服务器错误状态代码总数占总请求数的百分比	高级	详细状态代码	Y	sum(500InternalServerErrorCount)/sum(AllRequests)
503 服务不可用错误计数	503ServiceUnavailableErrorCount	503 服务不可用状态代码的总计数	高级	详细状态代码	否	-
503 服务不可用错误百分比	-	503 服务不可用状态代码总数占总请求数的百分比	高级	详细状态代码	Y	sum(503ServiceUnavailableErrorCount)/sum(AllRequests)

[1] 所有免费存储指标均在 Storage Lens 组级别可用。高级指标在 Storage Lens 组级别不可用。

[2] 规则计数指标和桶设置指标在前缀级别不可用。

通过控制台和 API 使用 Amazon S3 Storage Lens 存储统计管理工具

Amazon S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。您可以使用 S3 Storage Lens 存储统计管理工具指标生成摘要见解，例如，了解整个组织中有多少存储空间，或增长最快的桶和前缀是哪些。您还可以使用 S3 Storage Lens 存储统计管理工具指标来识别成本优化机会，实施数据保护和安全最佳实践，并提高应用程序工作负载的性能。例如，您可以识别没有 S3 生命周期规则的桶，使超过 7 天的未完成分段上传过期。您还可以识别未遵循数据保护最佳实践（例如使用 S3 复制或 S3 版本控制）的桶。S3 Storage Lens 存储统计管理工具还分析指标以提供上下文建议，您可以使用这些建议来优化存储成本并应用最佳实践来保护数据。

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 桶。

以下部分包含创建、更新和查看 S3 Storage Lens 存储统计管理工具配置以及执行与该功能相关的操作的示例。如果您在 AWS Organizations 中使用 S3 Storage Lens 存储统计管理工具，这些示例还涵盖了这些使用案例。在这些示例中，将任何变量值替换为符合您要求的变量值。

主题

- 在控制台中使用 Amazon S3 Storage Lens 存储统计管理工具
- 使用 AWS CLI 的 Amazon S3 Storage Lens 存储统计管理工具示例
- 使用适用于 Java 的 SDK 的 Amazon S3 Storage Lens 存储统计管理工具示例

在控制台中使用 Amazon S3 Storage Lens 存储统计管理工具

Amazon S3 Storage Lens 存储统计管理工具是一项云存储分析功能，您可以使用它在整个组织范围内了解对象存储的使用情况和活动。您可以使用 S3 Storage Lens 存储统计管理工具指标生成摘要见解，例如，了解整个组织中有多少存储空间，或增长最快的桶和前缀是哪些。您还可以使用 S3 Storage Lens 存储统计管理工具指标来识别成本优化机会，实施数据保护和安全最佳实践，并提高应用程序工作负载的性能。例如，您可以识别没有 S3 生命周期规则的桶，使超过 7 天的未完成分段上传过期。您

还可以识别未遵循数据保护最佳实践（例如使用 S3 复制或 S3 版本控制）的桶。S3 Storage Lens 存储统计管理工具还分析指标以提供上下文建议，您可以使用这些建议来优化存储成本并应用最佳实践来保护数据。

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、存储桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 存储桶。

> **Note**
>
> 对控制面板配置所做的任何更新最长可能需要 48 小时才能准确显示或可视化。

主题

创建和更新 Amazon S3 Storage Lens 存储统计管理工具控制面板

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、存储桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 存储桶。

Amazon S3 Storage Lens 存储统计管理工具默认控制面板是 default-account-dashboard。此控制面板由 Amazon S3 预配置，可帮助您在控制台上直观显示整个账户的聚合费用和高级指标的汇总见解和趋势。您无法修改原定设置控制面板的配置范围，但是您可以将指标选择从免费指标升级到付费的高级指标和建议，配置可选指标导出，甚至可以禁用原定设置控制面板。无法删除默认控制面板。

您还可以创建其他 S3 Storage Lens 存储统计管理工具自定义控制面板，其范围可以限定为涵盖 AWS Organizations 中您的组织，或者账户中的特定区域或桶。

创建 Amazon S3 Storage Lens 存储统计管理工具控制面板

使用以下步骤在 Amazon S3 控制台中创建 Amazon S3 Storage Lens 存储统计管理工具控制面板。

步骤 1：定义控制面板范围

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要切换到的区域。
3. 在导航窗格中，在 S3 Storage Lens 存储统计管理工具下选择控制面板。
4. 选择创建控制面板。
5. 在控制面板页面的常规部分中，执行以下操作：

   a. 查看控制面板的主区域。主区域是在其中存储此 Storage Lens 存储分析功能控制面板的配置和指标的 AWS 区域。

   b. 输入控制面板名称。

      控制面板名称必须少于 65 个字符，且不得包含特殊字符或空格。

      > **Note**
      >
      > 创建控制面板后，您无法更改此控制面板名称。

   c. 您可以选择将标签添加到控制面板。您可以使用标签管理控制面板的权限并跟踪 S3 Storage Lens 存储统计管理工具的成本。

      有关更多信息，请参阅《IAM 用户指南》中的使用资源标签控制访问以及 AWS Billing 用户指南中的 AWS 生成的成本分配标签。

      > **Note**
      >
      > 您最多可以在控制面板配置中添加 50 个标签。

6. 在控制面板范围部分中，执行以下操作：

   a. 选择希望 S3 Storage Lens 存储统计管理工具在控制面板中包括或排除的区域和桶。

   b. 在所选区域中选择希望 S3 Storage Lens 存储统计管理工具包括或排除的桶。您可以包括或排除桶，但不能同时包括和排除桶。创建组织级控制面板时，此选项不可用。

> **Note**
>
> - 您可以包括或排除区域和桶。在组织中跨成员账户创建组织级控制面板时，此选项仅限于“区域”。
> - 您最多可以选择 50 个桶来包括或排除。

步骤 2：配置指标选择

1. 在指标选择部分，选择要为此控制面板聚合的指标类型。

   - 要包括在桶级别聚合且可用于 14 天查询的免费指标，请选择 Free Metrics（免费指标）。
   - 要启用高级指标和其他高级选项，请选择 Advanced metrics and recommendations（高级指标和建议）。这些选项包括高级前缀聚合、Amazon CloudWatch 发布和上下文建议。数据可查询 15 个月。高级指标和建议需要额外付费。有关更多信息，请参阅 Amazon S3 定价。

     有关高级指标和免费指标的更多信息，请参阅指标选择。

2. 在 Advanced metrics and recommendations features（高级指标和建议功能）下方，选择您要启用的选项：

   - Advanced metrics（高级指标）
   - CloudWatch publishing（CloudWatch 发布）
   - Prefix aggregation（前缀聚合）

> **Important**
>
> 如果您为 S3 Storage Lens 存储统计管理工具配置启用了前缀聚合，那么前缀级别的指标将不会发布到 CloudWatch。只有桶、账户和企业级 S3 Storage Lens 存储统计管理工具指标才会发布到 CloudWatch。

3. 如果您启用了 Advanced metrics（高级指标），请选择要在 S3 Storage Lens 存储统计管理工具控制面板中显示的 Advanced metrics categories（高级指标类别）：

   - 活动指标
   - Detailed status code metrics（详细的状态代码指标）

- Advanced cost optimization metrics（高级成本优化指标）
- Advanced data protection metrics（高级数据保护指标）

有关指标类别的更多信息，请参阅指标类别。要获得指标的完整列表，请参阅 Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

4. 如果选择启用前缀聚合，请配置以下内容：

   a. 选择此控制面板的最小前缀阈值大小。

      例如，5% 的前缀阈值表示将聚合构成占桶的总存储大小 5% 或更大的前缀。

   b. 选择前缀名称。

      此设置指示评估前缀的最大级别数。前缀深度必须小于 10。

   c. 输入前缀分隔符字符。

      此值用来标识每个前缀级别。Amazon S3 中的原定设置值为 / 个字符，但是您的存储结构可能会使用其他分隔符字符。

（可选）步骤 3：导出控制面板的指标

1. 在 Metrics export（指标导出）部分中，要创建将每天放置在您选择的目标桶中的指标导出，请选择 Enable（启用）。

   指标导出为 CSV 或 Apache Parquet 格式。它代表的数据范围与 S3 Storage Lens 存储统计管理工具控制面板数据相同，但不包含建议。

2. 如果启用了指标导出，请选择每日指标导出的输出格式：CSV 或 Apache Parquet。

   Parquet 是 Hadoop 的开源文件格式，它以平面列格式存储嵌套数据。

3. 为您的指标导出选择目标 S3 桶。

   您可以在 S3 Storage Lens 存储统计管理工具控制面板的当前账户中选择桶。或者，如果您拥有目标桶权限和目标桶拥有者的账户 ID，则可以选择另一个 AWS 账户。

4. 选择目标 S3 桶（格式：`s3://bucket-name/prefix`）。

   桶必须在 S3 Storage Lens 存储统计管理工具控制面板的主区域中。S3 控制台向您显示 Amazon S3 将添加到目标桶策略的 Destination bucket permission（目标桶权限）。Amazon S3 将更新目标桶上的桶策略，以允许 S3 在该桶中放置数据。

5. （可选）要对指标导出启用服务器端加密，请选择 Specify an encryption key（指定加密密钥）。然后，选择加密类型：Amazon S3 托式管密钥（SSE-S3）或 AWS Key Management Service 密钥（SSE-KMS）。

   您可以在 Amazon S3 托管式密钥（SSE-S3）和 AWS Key Management Service（AWS KMS）密钥（SSE-KMS）之间进行选择。

6. （可选）要指定 AWS KMS 密钥，必须选择 KMS 密钥或输入密钥 Amazon 资源名称（ARN）。

   如果您选择客户托管式密钥，则必须在 AWS KMS 密钥策略中向 Amazon S3 Storage Lens 存储统计管理工具授予加密权限。有关更多信息，请参阅 使用 AWS KMS key 加密您的指标导出。

7. 请选择创建控制面板。

要进一步了解您的存储，您可以创建一个或多个 S3 Storage Lens 组并将它们附加到您的控制面板。S3 Storage Lens 组是基于前缀、后缀、对象标签、对象大小、对象龄期或这些筛选条件的组合的自定义对象筛选条件。

您可以使用 S3 Storage Lens 组获得对大型共享桶（例如数据湖）的精细可见性，从而做出更明智的业务决策。例如，您可以将存储使用情况细分到一个或多个桶中各个项目和成本中心的特定对象组，从而简化存储分配并优化成本报告。

要使用 S3 Storage Lens 组，您必须升级控制面板以使用高级指标和建议。有关 S3 Storage Lens 组的更多信息，请参阅 the section called “使用 S3 Storage Lens 组”。

更新 Amazon S3 Storage Lens 存储统计管理工具控制面板

使用以下步骤在 Amazon S3 控制台中更新 Amazon S3 Storage Lens 存储统计管理工具控制面板。

步骤 1：更新控制面板范围

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens（Storage Lens 存储统计管理工具）和 Dashboards（控制面板）。
3. 请选择要编辑的控制面板，然后选择 Edit（编辑）。

   此时将打开 Edit dashboard（编辑控制面板）页。

> **Note**
>
> 不能更改以下内容：
>
> - 控制面板名称
> - 主区域
> - 原定设置控制面板的控制面板范围，其范围限于整个账户的存储

4. （可选）在控制面板配置页面上的 General（常规）部分中，更新标签并向控制面板添加标签。

   您可以使用标签来管理控制面板的权限和跟踪 S3 Storage Lens 存储统计管理工具的成本。有关更多信息，请参阅《IAM 用户指南》中的使用资源标签控制访问以及 AWS Billing 用户指南中的 AWS 生成的成本分配标签。

   > **Note**
   >
   > 您最多可以在控制面板配置中添加 50 个标签。

5. 在控制面板范围部分中，执行以下操作：

   a. 更新希望 S3 Storage Lens 存储统计管理工具在控制面板中包含或排除的区域和桶。

      > **Note**
      >
      > - 您可以包括或排除区域和桶。在组织中跨成员账户创建组织级控制面板时，此选项仅限于“区域”。
      > - 您最多可以选择 50 个桶来包括或排除。

   b. 更新您希望 S3 Storage Lens 存储统计管理工具包含或排除的选定区域中的桶。您可以包括或排除桶，但不能同时包括和排除桶。创建组织级控制面板时，此选项不存在。

步骤 2：更新指标选择

1. 在指标选择部分，选择要为此控制面板聚合的指标类型。

   - 要包括在桶级别聚合且可用于 14 天查询的免费指标，请选择 Free Metrics（免费指标）。

- 要启用高级指标和其他高级选项，请选择 Advanced metrics and recommendations（高级指标和建议）。这些选项包括高级前缀聚合、Amazon CloudWatch 发布和上下文建议。数据可查询 15 个月。高级指标和建议需要额外付费。有关更多信息，请参阅 Amazon S3 定价。

  有关高级指标和免费指标的更多信息，请参阅指标选择。

2. 在 Advanced metrics and recommendations features（高级指标和建议功能）下方，选择您要启用的选项：

   - Advanced metrics（高级指标）
   - CloudWatch publishing（CloudWatch 发布）
   - Prefix aggregation（前缀聚合）

   > **Important**
   >
   > 如果您为 S3 Storage Lens 存储统计管理工具配置启用了前缀聚合，那么前缀级别的指标将不会发布到 CloudWatch。只有桶、账户和企业级 S3 Storage Lens 存储统计管理工具指标才会发布到 CloudWatch。

3. 如果您启用了 Advanced metrics（高级指标），请选择要在 S3 Storage Lens 存储统计管理工具控制面板中显示的 Advanced metrics categories（高级指标类别）：

   - 活动指标
   - Detailed status code metrics（详细的状态代码指标）
   - Advanced cost optimization metrics（高级成本优化指标）
   - Advanced data protection metrics（高级数据保护指标）

   有关指标类别的更多信息，请参阅指标类别。要获得指标的完整列表，请参阅 Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

4. 如果选择启用前缀聚合，请配置以下内容：

   a. 选择此控制面板的最小前缀阈值大小。

      例如，5% 的前缀阈值表示将聚合构成占桶的总存储大小 5% 或更大的前缀。

   b. 选择前缀名称。

      此设置指示评估前缀的最大级别数。前缀深度必须小于 10。

c. 输入前缀分隔符字符。

这是用来标识每个前缀级别的值。Amazon S3 中的原定设置值为 / 个字符，但是您的存储结构可能会使用其他分隔符字符。

（可选）步骤 3：导出控制面板的指标

1. 在 Metrics export（指标导出）部分中，要创建将每天放置在您选择的目标桶中的指标导出，请选择 Enable（启用）。要禁用指标导出，请选择 Disable（禁用）。

   指标导出为 CSV 或 Apache Parquet 格式。它代表的数据范围与 S3 Storage Lens 存储统计管理工具控制面板数据相同，但不包含建议。

2. 如果启用，请选择每日指标导出的输出格式：CSV 或 Apache Parquet。

   Parquet 是 Hadoop 的开源文件格式，它以平面列格式存储嵌套数据。

3. 为您的指标导出选择目标 S3 桶。

   您可以在 S3 Storage Lens 存储统计管理工具控制面板的当前账户中选择桶。或者，如果您拥有目标桶权限和目标桶拥有者的账户 ID，则可以选择另一个 AWS 账户。

4. 选择目标 S3 桶（格式：`s3://`*`bucket-name`*`/`*`prefix`*）。

   桶必须在 S3 Storage Lens 存储统计管理工具控制面板的主区域中。S3 控制台向您显示 Amazon S3 将添加到目标桶策略的 Destination bucket permission（目标桶权限）。Amazon S3 将更新目标桶上的桶策略，以允许 S3 在该桶中放置数据。

5. （可选）要对指标导出启用服务器端加密，请选择 Specify an encryption key（指定加密密钥）。然后，选择加密类型：Amazon S3 托式管密钥（SSE-S3）或 AWS Key Management Service 密钥（SSE-KMS）。

   您可以在 Amazon S3 托管式密钥（SSE-S3）和 AWS Key Management Service（AWS KMS）密钥（SSE-KMS）之间进行选择。

6. （可选）要指定 AWS KMS 密钥，必须选择 KMS 密钥或输入密钥 Amazon 资源名称（ARN）。在 AWS KMS 密钥下，通过以下方式之一指定您的 KMS 密钥：

   - 要从可用的 KMS 密钥列表中进行选择，请选择从您的 AWS KMS keys 密钥中进行选择，并从可用密钥的列表中选择您的 KMS 密钥。

AWS 托管式密钥 (aws/s3) 和您的客户自主管理型密钥都显示在此列表中。有关客户自主管理型密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的客户密钥和 AWS 密钥。

> **Note**
>
> 使用 S3 Storage Lens 存储统计管理工具进行的 SSE-KMS 加密不支持 AWS 托管式密钥（ aws/S3 ）。

- 要输入 KMS 密钥 ARN，请选择输入 AWS KMS key ARN，然后在显示的字段中输入您的 KMS 密钥 ARN。
- 要在 AWS KMS 控制台中创建新的客户自主管理型密钥，请选择创建 KMS 密钥。

  如果您选择客户托管式密钥，则必须在 AWS KMS 密钥策略中向 Amazon S3 Storage Lens 存储统计管理工具授予加密权限。有关更多信息，请参阅 使用 AWS KMS key 加密您的指标导出。

  有关创建 AWS KMS key 的更多信息，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。

7. 选择 Save changes（保存更改）。

要进一步了解您的存储，您可以创建一个或多个 S3 Storage Lens 组并将它们附加到您的控制面板。S3 Storage Lens 组是基于前缀、后缀、对象标签、对象大小、对象龄期或这些筛选条件的组合的自定义对象筛选条件。

您可以使用 S3 Storage Lens 组获得对大型共享桶（例如数据湖）的精细可见性，从而做出更明智的业务决策。例如，您可以将存储使用情况细分到一个或多个桶中各个项目和成本中心的特定对象组，从而简化存储分配并优化成本报告。

要使用 S3 Storage Lens 组，您必须升级控制面板以使用高级指标和建议。有关 S3 Storage Lens 组的更多信息，请参阅 the section called “使用 S3 Storage Lens 组”。

禁用或删除 Amazon S3 Storage Lens 存储统计管理工具控制面板

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、存储

桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 存储桶。

Amazon S3 Storage Lens 存储统计管理工具默认控制面板是 default-account-dashboard。此控制面板由 Amazon S3 预配置，可帮助您在控制台上直观显示整个账户的聚合费用和高级指标的汇总见解和趋势。您无法修改原定设置控制面板的配置范围，但是您可以将指标选择从免费指标升级到付费的高级指标和建议，配置可选指标导出，甚至可以禁用原定设置控制面板。无法删除默认控制面板。

您可以从 Amazon S3 控制台删除或禁用 Amazon S3 Storage Lens 存储统计管理工具控制面板。禁用或删除控制面板会阻止其将来生成指标。禁用的控制面板仍会保留其配置信息，以便重新启用后可以轻松恢复。禁用的控制面板将保留其历史数据，直到其不再可用于查询。

免费指标选择的数据可供查询 14 天，高级指标和建议选择的数据可供查询 15 个月。

禁用 Amazon S3 Storage Lens 存储统计管理工具控制面板

禁用 S3 Storage Lens 存储统计管理工具控制面板

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要禁用的控制面板，然后选择列表顶部的禁用。
4. 在确认页面上，通过在文本字段中输入控制面板名称来确认要禁用控制面板，然后选择确认。

删除 Amazon S3 Storage Lens 存储统计管理工具控制面板

> **Note**
>
> 您无法删除原定设置控制面板。但是，您可以将其禁用。删除您创建的控制面板之前，请考虑以下事项：
>
> - 作为删除控制面板的替代方法，您可以禁用控制面板，以便将来可以重新启用该控制面板。有关更多信息，请参阅 禁用 Amazon S3 Storage Lens 存储统计管理工具控制面板。
> - 删除控制面板将删除与其关联的所有配置设置。
> - 删除控制面板将使所有历史指标数据不可用。这些历史数据仍会保留 15 个月。如果您想再次访问此数据，请在与删除控制面板相同的主区域中以相同名称创建一个控制面板。

删除 S3 Storage Lens 存储统计管理工具控制面板

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 和 Dashboards（控制面板）。
3. 在控制面板列表中，选择要删除的控制面板，然后选择列表顶部的删除。
4. 在删除控制面板页面上，通过在文本字段中输入控制面板名称来确认要删除控制面板。然后，选择 Confirm（确认）。

使用 AWS Organizations 创建组织级控制面板

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、存储桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 存储桶。

Amazon S3 Storage Lens 存储统计管理工具默认控制面板是 default-account-dashboard。此控制面板由 Amazon S3 预配置，可帮助您在控制台上直观显示整个账户的聚合费用和高级指标的汇总见解和趋势。您无法修改原定设置控制面板的配置范围，但是您可以将指标选择从免费指标升级到付费的高级指标和建议，配置可选指标导出，甚至可以禁用原定设置控制面板。无法删除默认控制面板。

您还可以创建其他 S3 Storage Lens 存储统计管理工具控制面板，这些控制面板将重点放在企业中的特定 AWS 区域、S3 桶或其他 AWS 账户上。

S3 Storage Lens 存储统计管理工具控制面板提供了有关其存储范围的丰富信息资源。一个控制面板可以对超过 30 个指标进行可视化，这些指标显示了趋势和信息，包括存储摘要、成本效率、数据保护和活动。

Amazon S3 Storage Lens 存储统计管理工具可用于收集属于 AWS Organizations 层次结构的所有账户的存储指标和使用情况数据。为此，您必须使用 AWS Organizations，并且必须使用您的 AWS Organizations 管理账户启用 S3 Storage Lens 存储统计管理工具可信访问权限。

启用可信访问权限后，您可以为组织中的账户添加委托管理员访问权限。然后，这些账户可以为 S3 Storage Lens 存储统计管理工具创建组织范围的控制面板和配置。有关启用可信访问权限的更多信息，请参阅《AWS Organizations 用户指南》中的 Amazon S3 Lens 和 AWS Organizations。

以下控制台控件仅适用于 AWS Organizations 管理账户。

在组织中为 S3 Storage Lens 存储统计管理工具启用可信访问权限

启用可信访问权限允许 Amazon S3 Storage Lens 存储统计管理工具通过 AWS Organizations API 操作访问您的 AWS Organizations 层次结构、成员资格和结构。S3 Storage Lens 存储统计管理工具成为整个组织的结构的可信服务。每当创建控制面板配置时，此工具都可以在组织的管理或委派管理员账户中创建服务相关角色。

服务相关角色向 S3 Storage Lens 存储统计管理工具授予权限，以描述组织、列出账户、验证组织的服务访问列表以及获取组织的委托管理员权限。这允许 S3 Storage Lens 存储统计管理工具收集组织中账户内控制面板的跨账户存储使用情况和活动指标。

有关更多信息，请参阅 将服务相关角色用于 Amazon S3 Storage Lens 存储统计管理工具。

> **Note**
>
> - 只能由管理账户启用可信访问权限。
> - 只有管理账户和委托管理员才能为您的组织创建 S3 Storage Lens 存储统计管理工具控制面板或配置。

启用 S3 Storage Lens 存储统计管理工具以获得可信访问权限

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，依次选择 Storage Lens 和 Organization settings（组织设置）。
3. 在组织访问权限中，选择编辑。

   此时将打开组织访问权限页面。在这里，您可以为 S3 Storage Lens 存储统计管理工具启用可信访问权限。这允许您和作为委托管理员添加的任何其他账户持有人为组织中的所有账户和存储空间创建控制面板。

禁用组织中的 S3 Storage Lens 存储统计管理工具可信访问权限

禁用可信访问权限将限制 S3 Storage Lens 存储统计管理工具只能在账户级别运行。每个账户持有人所看到的 S3 Storage Lens 存储统计管理工具优势仅限于其账户范围，而不是组织范围。任何需要可信访问权限的控制面板都将不再更新，但这些控制面板将能够在数据可供查询的相应时间段内查询它们的历史数据。

以委派管理员身份删除账户会将账户拥有者的 S3 Storage Lens 存储统计管理工具控制面板指标访问权限限制为仅在账户级别起作用。任何所创建组织的控制面板将不再被更新，但它们将能够 period that data is available for queries（在数据可供查询的相应时间段内）查询它们的历史数据。

> **Note**
>
> - 禁用可信访问权限还会自动禁用所有组织级控制面板，因为 S3 Storage Lens 存储统计管理工具将不再具有对组织账户的可信访问权限以收集和聚合存储指标。
> - 管理员和委托管理员账户仍然可以查看这些禁用的控制面板的历史数据，并在可用期间查询此数据。

为 S3 Storage Lens 存储统计管理工具禁用可信访问权限

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，依次选择 Storage Lens 和 Organization settings（组织设置）。
3. 在组织访问权限中，选择编辑。

   此时将打开组织访问权限页面。在这里，您可以为 S3 Storage Lens 存储统计管理工具禁用可信访问权限。

为 S3 Storage Lens 存储统计管理工具注册委托管理员

启用可信访问权限后，您可以注册组织中账户的委托管理员访问权限。当账户注册为委托管理员后，该账户将获得访问所有只读 AWS Organizations API 操作的授权。这就提供了对您组织成员和结构的可见性，以便他们可以代表您创建 S3 Storage Lens 存储统计管理工具控制面板。

为 S3 Storage Lens 存储统计管理工具注册委托管理员

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，依次选择 Storage Lens 和 Organization settings（组织设置）。
3. 在委托访问权限部分中，针对账户选择添加账户。

   将打开委托管理员访问权限页面。在这里，您可以添加 AWS 账户 ID 作为委托管理员，以便为企业中的所有账户和存储创建企业级控制面板。

为 S3 Storage Lens 存储统计管理工具取消注册委托管理员

您可以取消注册委托管理员对组织中账户的访问权限。当账户被取消注册委托管理员时，该账户将失去能够看到组织成员和结构的所有只读 AWS Organizations API 操作的访问授权。

> **Note**
>
> - 取消注册委托管理员还会自动禁用委托管理员创建的所有组织级控制面板。
> - 委托管理员账户仍然可以根据查询数据的可用时间查看这些禁用控制面板的历史数据。

取消注册具有委托管理员访问权限的账户

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，依次选择 Storage Lens 和 Organization settings（组织设置）。
3. 在具有委托访问权限的账户部分中，选择要取消注册的账户 ID，然后选择删除。

使用 AWS CLI 的 Amazon S3 Storage Lens 存储统计管理工具示例

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 桶。有关更多信息，请参阅使用 Amazon S3 Storage Lens 存储统计管理工具评估存储活动和使用情况。

以下示例说明如何将 S3 Storage Lens 存储统计管理工具与 AWS Command Line Interface 结合使用。

主题

- 使用 Amazon S3 Storage Lens 存储统计管理工具的帮助程序文件
- 将 Amazon S3 Storage Lens 存储统计管理工具配置与 AWS CLI 结合使用
- 通过 AWS CLI 结合使用 Amazon S3 Storage Lens 存储统计管理工具与 AWS Organizations 示例

使用 Amazon S3 Storage Lens 存储统计管理工具的帮助程序文件

为您的示例使用以下 JSON 文件及其关键输入。

JSON 格式的 S3 Storage Lens 存储统计管理工具示例配置

Example **config.json**

config.json 文件包含 S3 Storage Lens 存储统计管理工具组织级高级指标和建议配置的详细信息。要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

> **Note**
>
> 高级指标和建议需支付额外费用。有关详细信息，请参阅高级指标和建议。

```
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true
      },
      "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
      },
      "AdvancedCostOptimizationMetrics": {
```

```
        "IsEnabled":true
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true,
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
configurations.
      "arn:aws:s3:::source_bucket1"
    ]
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::destination-bucket-name", // The destination bucket for your
metrics export must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true
    }
  }
```

```
}
```

JSON 格式的 S3 Storage Lens 存储统计管理工具示例配置（带 Storage Lens 组）

Example **config.json**

config.json 文件包含在使用 Storage Lens 组时，要应用于 Storage Lens 存储统计管理工具配置的详细信息。要使用该示例，请将 *user input placeholders* 替换为您自己的信息。

要将所有 Storage Lens 组附加到您的控制面板，请使用以下语法更新 Storage Lens 存储统计管理工具配置：

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
    "StorageLensGroupLevel": {},
  "IsEnabled": true
}
```

要在 Storage Lens 存储统计管理工具控制面板配置中仅包含两个 Storage Lens 组（*slg-1* 和 *slg-2*），请使用以下语法：

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
```

```
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Include": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

要仅将某些 Storage Lens 组排除在您的控制面板配置之外，请使用以下语法：

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Exclude": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
```

```
    },
  "IsEnabled": true
}
```

JSON 格式的 S3 Storage Lens 存储统计管理工具示例标签配置

Example **tags.json**

tags.json 文件包含要应用于 S3 Storage Lens 存储统计管理工具配置的标签。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
[
    {
        "Key": "key1",
        "Value": "value1"
    },
    {
        "Key": "key2",
        "Value": "value2"
    }
]
```

S3 Storage Lens 存储统计管理工具示例配置 IAM 权限

Example **permissions.json** – 特定控制面板名称

此示例策略显示具有指定的特定控制面板名称的 S3 Storage Lens 存储统计管理工具 IAM permissions.json 文件。将 *value1*、*us-east-1*、*your-dashboard-name* 和 *example-account-id* 替换为您自己的值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/your-
dashboard-name"
        }
    ]
}
```

Example **permissions.json** – 没有特定的控制面板名称

此示例策略显示了未指定特定控制面板名称的 S3 Storage Lens 存储统计管理工具 IAM permissions.json 文件。将 *value1*、*us-east-1* 和 *example-account-id* 替换为您自己的值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:example-account-id:storage-lens/*"
        }
    ]
}
```

将 Amazon S3 Storage Lens 存储统计管理工具配置与 AWS CLI 结合使用

您可以使用 AWS CLI 列出、创建、删除、获取、标记和更新 S3 Storage Lens 存储统计管理工具配置。以下示例使用帮助程序 JSON 文件进行关键输入。要使用这些示例，请将 *user input placeholders* 替换为您自己的信息。

创建 S3 Storage Lens 存储统计管理工具配置

Example 创建 S3 Storage Lens 存储统计管理工具配置

```
aws s3control put-storage-lens-configuration --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --storage-lens-
configuration=file://./config.json --tags=file://./tags.json
```

创建没有标签的 S3 Storage Lens 存储统计管理工具配置

Example 创建没有标签的 S3 Storage Lens 存储统计管理工具配置

```
aws s3control put-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json
```

获取 S3 Storage Lens 存储统计管理工具配置

Example 获取 S3 Storage Lens 存储统计管理工具配置

```
aws s3control get-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1
```

列出没有下一个令牌的 S3 Storage Lens 存储统计管理工具配置

Example 列出没有下一个令牌的 S3 Storage Lens 存储统计管理工具配置

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-
east-1
```

列出 S3 Storage Lens 存储统计管理工具配置

Example 列出 S3 Storage Lens 存储统计管理工具配置

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-
east-1 --next-token=abcdefghij1234
```

删除 S3 Storage Lens 存储统计管理工具配置

Example 删除 S3 Storage Lens 存储统计管理工具配置

```
aws s3control delete-storage-lens-configuration --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

将标签添加到 S3 Storage Lens 存储统计管理工具配置

Example 将标签添加到 S3 Storage Lens 存储统计管理工具配置

```
aws s3control put-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id --tags=file://./tags.json
```

获取 S3 Storage Lens 存储统计管理工具配置标签

Example 获取 S3 Storage Lens 存储统计管理工具配置标签

```
aws s3control get-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

删除 S3 Storage Lens 存储统计管理工具配置标签

Example 删除 S3 Storage Lens 存储统计管理工具配置标签

```
aws s3control delete-storage-lens-configuration-tagging --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

通过 AWS CLI 结合使用 Amazon S3 Storage Lens 存储统计管理工具与 AWS Organizations 示例

使用 Amazon S3 Storage Lens 存储统计管理工具收集属于 AWS Organizations 层次结构的所有账户的存储指标和使用情况数据。有关更多信息，请参阅将 Amazon S3 Storage Lens 存储统计管理工具与 AWS Organizations 结合使用。

为 S3 Storage Lens 存储统计管理工具启用组织可信访问权限

Example 为 S3 Storage Lens 存储统计管理工具启用组织可信访问权限

```
aws organizations enable-aws-service-access --service-principal storage-lens.s3.amazonaws.com
```

为 S3 Storage Lens 存储统计管理工具禁用组织可信访问权限

Example 为 S3 Storage Lens 存储统计管理工具禁用组织可信访问权限

```
aws organizations disable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

为 S3 Storage Lens 存储统计管理工具注册组织委托管理员

Example 为 S3 Storage Lens 存储统计管理工具注册组织委托管理员

要使用此示例，请将 *111122223333* 替换为相应的 AWS 账户 ID。

```
aws organizations register-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

为 S3 Storage Lens 存储统计管理工具取消注册组织委托管理员

Example 为 S3 Storage Lens 存储统计管理工具取消注册组织委托管理员

要使用此示例，请将 *111122223333* 替换为相应的 AWS 账户 ID。

```
aws organizations deregister-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

使用适用于 Java 的 SDK 的 Amazon S3 Storage Lens 存储统计管理工具示例

S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 桶。有关更多信息，请参阅使用 Amazon S3 Storage Lens 存储统计管理工具评估存储活动和使用情况。

以下示例说明如何将 S3 Storage Lens 存储统计管理工具与 AWS SDK for Java 结合使用。

主题

- 使用适用于 Java 的开发工具包使用 Amazon S3 Storage Lens 存储统计管理工具配置

使用适用于 Java 的开发工具包使用 Amazon S3 Storage Lens 存储统计管理工具配置

您可以使用适用于 Java 的开发工具包列出、创建、获取和更新 S3 Storage Lens 存储统计管理工具配置。以下示例使用帮助程序 JSON 文件进行关键输入。

主题

创建和更新 S3 Storage Lens 存储统计管理工具配置

Example 创建和更新 S3 Storage Lens 存储统计管理工具配置

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
```

```
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
```

```
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);
```

```
            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

删除 S3 Storage Lens 存储统计管理工具配置

Example 删除 S3 Storage Lens 存储统计管理工具配置

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;
```

```
import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfiguration(new
 DeleteStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

获取 S3 Storage Lens 存储统计管理工具配置

Example 获取 S3 Storage Lens 存储统计管理工具配置

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationResult;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
```

```
import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final StorageLensConfiguration configuration =
                    s3ControlClient.getStorageLensConfiguration(new
 GetStorageLensConfigurationRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getStorageLensConfiguration();

            System.out.println(configuration.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

列出 S3 Storage Lens 存储统计管理工具配置

Example 列出 S3 Storage Lens 存储统计管理工具配置

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
```

```
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationEntry;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationsRequest;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class ListDashboard {

    public static void main(String[] args) {
        String sourceAccountId = "Source Account ID";
        String nextToken = "nextToken";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final List<ListStorageLensConfigurationEntry> configurations =
                    s3ControlClient.listStorageLensConfigurations(new
 ListStorageLensConfigurationsRequest()
                            .withAccountId(sourceAccountId)
                            .withNextToken(nextToken)
                    ).getStorageLensConfigurationList();

            System.out.println(configurations.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

将标签添加到 S3 Storage Lens 存储统计管理工具配置

Example 将标签添加到 S3 Storage Lens 存储统计管理工具配置

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.PutStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class PutDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";

        try {
            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfigurationTagging(new
 PutStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
```

```
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

获取 S3 Storage Lens 存储统计管理工具配置标签

Example 获取 S3 Storage Lens 存储统计管理工具配置标签

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;
import
 com.amazonaws.services.s3control.model.GetStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            final List<StorageLensTag> s3Tags = s3ControlClient
```

```
                    .getStorageLensConfigurationTagging(new
 GetStorageLensConfigurationTaggingRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getTags();

            System.out.println(s3Tags.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

删除 S3 Storage Lens 存储统计管理工具配置标签

Example 删除 S3 Storage Lens 存储统计管理工具配置标签

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationTaggingRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
```

```
                    .build();

            s3ControlClient.deleteStorageLensConfigurationTagging(new
 DeleteStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用高级指标和建议更新原定设置 S3 Storage Lens 存储统计管理工具配置

Example 使用高级指标和建议更新原定设置 S3 Storage Lens 存储统计管理工具配置

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
```

```
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateDefaultConfigWithPaidFeatures {

    public static void main(String[] args) {
        String configurationId = "default-account-dashboard"; // This configuration ID
 cannot be modified.
        String sourceAccountId = "Source Account ID";

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

> **Note**
>
> 高级指标和建议需支付额外费用。有关详细信息，请参阅高级指标和建议。

将 Storage Lens 组附加到 S3 Storage Lens 存储统计管理工具控制面板

Example 将所有 Storage Lens 组附加到控制面板

以下 Java 版 SDK 示例将账户 *111122223333* 中的所有 Storage Lens 组附加到 *DashBoardConfigurationId* 控制面板：

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
```

```
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWithStorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel();

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 将两个 Storage Lens 组附加到控制面板

以下 AWS SDK for Java 示例将两个 Storage Lens 组（*StorageLensGroupName1* 和 *StorageLensGroupName2*）附加到 *ExampleDashboardConfigurationId* 控制面板。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);
```

```
            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example 附加除排除项之外的所有 Storage Lens 组

以下 Java 版 SDK 示例将所有 Storage Lens 组附加到 *ExampleDashboardConfigurationId* 控制面板，但不包括指定的两个组（*StorageLensGroupName1* 和 *StorageLensGroupName2*）：

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
```

```
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroupsExcluded {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
```

```
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 SDK for Java 将 Amazon S3 Storage Lens 存储统计管理工具与 AWS Organizations 示例结合使用

使用 Amazon S3 Storage Lens 存储统计管理工具收集属于 AWS Organizations 层次结构的所有账户的存储指标和使用情况数据。有关更多信息，请参阅将 Amazon S3 Storage Lens 存储统计管理工具与 AWS Organizations 结合使用。

主题

- 为 S3 Storage Lens 存储统计管理工具启用组织可信访问权限
- 为 S3 Storage Lens 存储统计管理工具禁用组织可信访问权限
- 为 S3 Storage Lens 存储统计管理工具注册组织委托管理员
- 为 S3 Storage Lens 存储统计管理工具取消注册组织委托管理员

为 S3 Storage Lens 存储统计管理工具启用组织可信访问权限

Example 为 S3 Storage Lens 存储统计管理工具启用组织可信访问权限

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.EnableAWSServiceAccessRequest;
```

```
public class EnableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.enableAWSServiceAccess(new
 EnableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

为 S3 Storage Lens 存储统计管理工具禁用组织可信访问权限

Example 为 S3 Storage Lens 存储统计管理工具禁用组织可信访问权限

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.DisableAWSServiceAccessRequest;

public class DisableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";
```

```
 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            // Make sure to remove any existing delegated administrator for S3 Storage
 Lens
            // before disabling access; otherwise, the request will fail.
            organizationsClient.disableAWSServiceAccess(new
 DisableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

为 S3 Storage Lens 存储统计管理工具注册组织委托管理员

Example 为 S3 Storage Lens 存储统计管理工具注册组织委托管理员

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.RegisterDelegatedAdministratorRequest;

public class RegisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
```

```
  try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.registerDelegatedAdministrator(new
 RegisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

为 S3 Storage Lens 存储统计管理工具取消注册组织委托管理员

Example 为 S3 Storage Lens 存储统计管理工具取消注册组织委托管理员

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.DeregisterDelegatedAdministratorRequest;

public class DeregisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
```

```
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.deregisterDelegatedAdministrator(new
 DeregisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

使用 S3 Storage Lens 组

Amazon S3 Storage Lens 存储统计管理工具组使用基于对象元数据的自定义筛选条件聚合指标。Storage Lens 组可帮助您深入了解数据的特征，例如按龄期划分的对象分布、最常见的文件类型等。例如，您可以按对象标签筛选指标以识别增长最快的数据集，或者根据对象大小和龄期可视化存储，为制定存储存档策略提供信息。因此，Amazon S3 Storage Lens 存储统计管理工具组可以帮助您更好地了解和优化 S3 存储。

使用 Storage Lens 组时，您可以使用对象元数据（例如前缀、后缀、对象标签、对象大小或对象龄期）来分析和筛选 S3 Storage Lens 存储统计管理工具指标。您也可以组合应用这些筛选条件。将 Storage Lens 组附加到 S3 Storage Lens 存储统计管理工具控制面板后，您可以直接在控制面板中查看 Amazon S3 Storage Lens 存储统计管理工具组聚合的 S3 Storage Lens 存储统计管理工具指标。

例如，您还可以按对象大小或龄段筛选指标，以确定存储的哪一部分由小对象组成。然后，您可以将此信息与 S3 Intelligent-Tiering 或 S3 生命周期结合使用，将小对象过渡到不同的存储类，以实现成本和存储优化。

主题

- S3 Storage Lens 组工作原理
- 使用 Storage Lens 组

S3 Storage Lens 组工作原理

您可以通过 Storage Lens 组，使用基于对象元数据的自定义筛选条件来聚合指标。定义自定义筛选条件时，可以使用前缀、后缀、对象标签、对象大小、对象期龄或这些自定义筛选条件的组合。在创建 Storage Lens 组期间，您还可以包括一个或多个筛选条件。要指定多个筛选条件，请使用 `And` 或 `Or` 逻辑运算符。

创建和配置 Storage Lens 组时，该 Storage Lens 组本身将充当要将该组附加到的控制面板中的自定义筛选条件。然后，在控制面板中，您可以使用 Storage Lens 组筛选条件，根据您在组中定义的自定义筛选条件来获取存储指标。

要在 S3 Storage Lens 存储统计管理工具控制面板中查看 Storage Lens 组的数据，您必须在创建该组后将其附加到控制面板。将 Storage Lens 组关联到 Storage Lens 存储统计管理工具控制面板后，您的控制面板将在 48 小时内收集存储使用情况指标。然后，您可以在 Storage Lens 存储统计管理工具控制面板中可视化这些数据，也可以通过指标导出将其导出。如果您忘记将 Storage Lens 组附加到控制面板，则不会在任何地方捕获或显示您的 Storage Lens 组数据。

> **Note**
>
> - 创建 S3 Storage Lens 组时，您将创建一个 AWS 资源。因此，每个 Storage Lens 组都有自己的 Amazon 资源名称 (ARN)，您可以在将 Storage Lens 组附加到 S3 Storage Lens 存储统计管理工具控制面板或将其从该控制面板排除时指定该名称。
> - 如果您的 Storage Lens 组未附加到控制面板，则创建 Storage Lens 组不会产生任何额外费用。
> - S3 Storage Lens 存储统计管理工具聚合了所有匹配的 Storage Lens 组下某个对象的使用情况指标。因此，如果某个对象与两个或更多 Storage Lens 组的筛选条件相匹配，则在整个存储使用情况中，您会看到同一对象的重复计数。

您可以在指定的（支持的 AWS 区域 列表中的）主区域中创建账户级别的 Storage Lens 组。然后，您可以将您的 Storage Lens 组附加到多个 Storage Lens 存储统计管理工具控制面板，前提是这些控制面板位于同一个 AWS 账户和主区域中。您最多可以为每个 AWS 账户中的每个主区域创建 50 个 Storage Lens 组。

您可以使用 Amazon S3 控制台、AWS Command Line Interface (AWS CLI)、AWS SDK 或 Amazon S3 REST API 创建和管理 S3 Storage Lens 组。

主题

查看 Storage Lens 组聚合指标

您可以通过将 Storage Lens 组附加到控制面板来查看这些组的聚合指标。要附加的 Storage Lens 组必须位于控制面板账户中指定的主区域内。

要将 Storage Lens 组附加到控制面板，您必须在控制面板配置的 Storage Lens 组聚合部分指定该组。如果您有多个 Storage Lens 组，则可以筛选 Storage Lens 组聚合结果，使其根据需要仅包括或排除某些组。有关将组附加到控制面板的更多信息，请参阅 the section called “附加或移除 Storage Lens 组”。

附加组后，您将在 48 小时内在控制面板中看到新增的 Storage Lens 组聚合数据。

> **Note**
>
> 要查看您的 Storage Lens 组的聚合指标，您必须将该组附加到 S3 Storage Lens 存储统计管理工具控制面板。

Storage Lens 组权限

Storage Lens 组需要 AWS Identity and Access Management (IAM) 中的某些权限才能授权访问 S3 Storage Lens 组操作。要授予这些权限，您可以使用基于身份的 IAM policy。您可以将此策略附加到 IAM 用户、组或角色，以授予其权限。此类权限可能包括创建或删除 Storage Lens 组、查看其配置或管理其标签。

您授予权限的 IAM 用户或角色必须属于创建或拥有 Storage Lens 组的账户。

要使用 Storage Lens 组和查看您的 Storage Lens 组指标，您必须首先拥有使用 S3 Storage Lens 存储统计管理工具的相应权限。有关更多信息，请参阅the section called “S3 Storage Lens 存储统计管理工具权限”。

要创建和管理 S3 Storage Lens 组，您必须拥有以下 IAM 权限，具体取决于您要执行的操作：

操作	IAM 权限
新建 Storage Lens 组	`s3:CreateStorageLensGroup`
使用标签新建 Storage Lens 组	`s3:CreateStorageLensGroup` ， `s3:TagResource`
更新现有 Storage Lens 组	`s3:UpdateStorageLensGroup`
返回 Storage Lens 组配置的详细信息	`s3:GetStorageLensGroup`
列出主区域中的所有 Storage Lens 组	`s3:ListStorageLensGroups`
删除 Storage Lens 组	`s3:DeleteStorageLensGroup`
列出已添加到 Storage Lens 组的标签	`s3:ListTagsForResource`
为现有 Storage Lens 组添加或更新 Storage Lens 组标签	`s3:TagResource`
从 Storage Lens 组删除标签	`s3:UntagResource`

以下是如何在创建 Storage Lens 组的账户中配置 IAM policy 的示例。要使用此策略，请将 *`us-east-1`* 替换为您的 Storage Lens 组所在的主区域。将 *`111122223333`* 替换为您的 AWS 账户 ID，然后将 *`example-storage-lens-group`* 替换为您的 Storage Lens 组的名称。要将这些权限应用于所有 Storage Lens 组，请将 *`example-storage-lens-group`* 替换为 *`*`*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EXAMPLE-Statement-ID",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:CreateStorageLensGroup",
                "s3:UpdateStorageLensGroup",
                "s3:GetStorageLensGroup",
                "s3:ListStorageLensGroups",
                "s3:DeleteStorageLensGroup,
                "s3:TagResource",
                "s3:UntagResource",
                "s3:ListTagsForResource"
                ],
            "Resource": "arn:aws:s3:us-east-1:111122223333:storage-lens-group/example-storage-lens-group"
        }
    ]
}
```

有关 S3 Storage Lens 存储统计管理工具权限的更多信息，请参阅 Amazon S3 Storage Lens 存储统计管理工具权限。有关 IAM policy 语言的更多信息，请参阅 Amazon S3 中的策略和权限。

Storage Lens 组配置

S3 Storage Lens 组名称

我们建议您使用 Storage Lens 组的用途为其命名，这样您就可以轻松确定要将哪些组附加到控制面板。要将 Storage Lens 组附加到控制面板，您必须在控制面板配置的 Storage Lens 组聚合部分指定该组。

账户中的 Storage Lens 组名称必须唯一。长度不得超过 64 个字符，只能包含字母（a-z、A-Z）、数字 (0-9)、连字符 (-) 和下划线 (_)。

主区域

主区域是创建和维护 Storage Lens 组的 AWS 区域。您的 Storage Lens 组是在与 Amazon S3 Storage Lens 存储统计管理工具控制面板相同的主区域创建的。Storage Lens 组配置和指标也存储在该区域中。您最多可以在主区域中创建 50 个 Storage Lens 组。

创建 Storage Lens 组后，您无法编辑主区域。

范围

要将对象包含在 Storage Lens 组中，这些对象必须在 Amazon S3 Storage Lens 存储统计管理工具控制面板的范围内。Storage Lens 存储统计管理工具控制面板的范围由您在 S3 Storage Lens 存储统计管理工具控制面板配置的控制面板范围中包含的桶决定。

您可以为对象使用不同的筛选条件来定义 Storage Lens 组的范围。要在 S3 Storage Lens 存储统计管理工具控制面板中查看这些 Storage Lens 组指标，对象必须与您在 Storage Lens 组中包含的筛选条件相匹配。例如，假设您的 Storage Lens 组要包括带有前缀 `marketing` 和后缀 `.png` 的对象，但没有对象与这些条件相匹配。在这种情况下，在您的每日指标导出中不会生成此 Storage Lens 组的指标，并且您的控制面板中也不会显示该组的指标。

筛选条件

您可以在 S3 Storage Lens 组中使用以下筛选条件：

- 前缀 – 指定要包括的对象的前缀，该前缀是对象键名称开头的一串字符。例如，如果前缀筛选条件的值为 `images`，则将包括带有以下任何前缀的对象：`images/`、`images-marketing` 和 `images/production`。前缀的最大长度是 1024 个字节。
- 后缀 – 指定要包括的对象的后缀（例如 `.png`、`.jpeg` 或 `.csv`）。后缀的最大长度是 1024 个字节。
- 对象标签 – 指定要用于筛选的对象标签列表。标签键不能超过 128 个 Unicode 字符，标签值不能超过 256 个 Unicode 字符。请注意，如果将对象标签值字段留空，则 S3 Storage Lens 组仅将该对象与标签值也为空的其它对象进行匹配。
- 龄期 – 指定要包括的对象的对象龄期范围（以天为单位）。仅支持整数。
- 大小 – 指定要包括的对象的对象大小范围（以字节为单位）。仅支持整数。允许的最大值为 5 TB。

Storage Lens 组对象标签

您可以创建最多包含 10 个对象标签筛选条件的 Storage Lens 组。以下示例包含两个对象标签键值对作为名为 *`Marketing-Department`* 的 Storage Lens 组的筛选条件。要使用此示例，请使用您的组名称替换 *`Marketing-Department`*，并将 *`object-tag-key-1`*、*`object-tag-value-1`* 等替换为要用于筛选的对象标签键值对。

```
{
    "Name": "Marketing-Department",
    "Filter": {
     "MatchAnyTag":[
                {
                    "Key": "object-tag-key-1",
                    "Value": "object-tag-value-1"
                },
                {
                    "Key": "object-tag-key-2",
                    "Value": "object-tag-value-2"
```

```
                }
            ]
    }
}
```

逻辑运算符（**And** 或 **Or**）

要在 Storage Lens 组中包含多个筛选条件，可以使用逻辑运算符（And 或 Or）。在以下示例中，名为 *Marketing-Department* 的 Storage Lens 组使用 And 运算符包含 Prefix、ObjectAge 和 ObjectSize 筛选条件。由于使用 And 运算符，因此只有与所有这些筛选条件匹配的对象才会包括在 Storage Lens 组的范围内。

要使用此示例，请将 *user input placeholders* 替换为要用于筛选的值。

```
{
    "Name": "Marketing-Department",
    "Filter": {
        "And": {
            "MatchAnyPrefix": [
                "prefix-1",
                "prefix-2",
                "prefix-3/sub-prefix-1"
            ],
             "MatchObjectAge": {
                "DaysGreaterThan": 10,
                "DaysLessThan": 60
            },
            "MatchObjectSize": {
                "BytesGreaterThan": 10,
                "BytesLessThan": 60
            }
        }
    }
}
```

> **Note**
>
> 如果要包括与任何筛选条件相匹配的对象，请将此示例中的 And 逻辑运算符替换为 Or 逻辑运算符。

AWS 资源标签

每个 S3 Storage Lens 组都视为一个 AWS 资源，有自己的 Amazon 资源名称 (ARN)。因此，在配置 Storage Lens 组时，可以选择向该组添加 AWS 资源标签。最多可以为每个 Storage Lens 组添加 50 个标签。要创建带标签的 Storage Lens 组，您必须拥有 `s3:CreateStorageLensGroup` 和 `s3:TagResource` 权限。

您可以使用 AWS 资源标签根据部门、业务线或项目对资源进行分类。如果您有许多相同类型的资源，这样做会非常有用。通过应用标签，您可以根据分配给特定 Storage Lens 组的标签快速识别该组。您还可以使用标签对成本进行跟踪和分配。

此外，当您向 Storage Lens 组添加 AWS 资源标签时，将激活基于属性的访问权限控制 (ABAC)。ABAC 是一种基于属性（在本文中，也就是标签）定义权限的授权策略。您还可以在 IAM policy 中使用指定资源标签的条件，从而控制对 AWS 资源的访问。

您可以修改标签的键和值，还可以随时删除资源的标签。此外，请注意以下限制：

- 标签键和标签值区分大小写。
- 如果您添加的标签的值与该实例上现有标签的值相同，新的值就会覆盖旧值。
- 如果删除资源，资源的所有标签也会被删除。
- 不要在 AWS 资源标签中包含隐私或敏感数据。
- 不支持系统标签（或标签键以 `aws:` 开头的标签）。
- 每个标签键的长度不能超过 128 个字符。每个标签值的长度不能超过 256 个字符。

Storage Lens 组指标导出

S3 Storage Lens 组指标包含在 Storage Lens 组所附加到的控制面板的 Amazon S3 Storage Lens 存储统计管理工具指标导出中。有关 Storage Lens 指标导出功能的一般信息，请参阅 使用数据导出查看 Amazon S3 Storage Lens 存储统计管理工具指标。

Storage Lens 组的指标导出包括您将 Storage Lens 组附加到的控制面板范围内的任何 S3 Storage Lens 指标。导出还包括 Storage Lens 组的其它指标数据。

创建 Storage Lens 组后，您的指标导出每天都会发送到您在为组所附加到的控制面板配置指标导出时选择的桶。您最长可能需要 48 小时才能收到第一份指标导出。

要在每日导出中生成指标，对象必须与您在 Storage Lens 组中包含的筛选条件相匹配。如果没有对象与您在 Storage Lens 组中包含的筛选条件相匹配，则不会生成任何指标。但是，如果某个对象与两个或更多 Storage Lens 组相匹配，则当该对象出现在指标导出中时，该对象将针对每个组单独列出。

您可以通过在控制面板指标导出的 `record_type` 列中查找以下值之一来识别 Storage Lens 组的指标：

- `STORAGE_LENS_GROUP_BUCKET`
- `STORAGE_LENS_GROUP_ACCOUNT`

`record_value` 列显示 Storage Lens 组的资源 ARN（例如 `arn:aws:s3:us-east-1:111122223333:storage-lens-group/Marketing-Department`）。

使用 Storage Lens 组

Amazon S3 Storage Lens 存储统计管理工具组使用基于对象元数据的自定义筛选条件聚合指标。您可以使用前缀、后缀、对象标签、对象大小或对象龄期来分析和筛选 S3 Storage Lens 存储统计管理工具指标。使用 Amazon S3 Storage Lens 存储统计管理工具组，您还可以在 Amazon S3 桶内以及跨多个桶，对使用情况进行分类。因此，您将能够更好地了解和优化 S3 存储。

要开始可视化 Storage Lens 组的数据，必须先将 Storage Lens 组附加到 S3 Storage Lens 存储统计管理工具控制面板。如果您需要在控制面板中管理 Storage Lens 组，可以编辑控制面板配置。要查看您的账户下有哪些 Storage Lens 组，您可以将其列出。要查看哪些 Storage Lens 组已附加到控制面板，您可以随时查看控制面板中的 Storage Lens 组选项卡。要查看或更新现有 Storage Lens 组的范围，可以查看其详细信息。您还可以永久删除 Storage Lens 组。

要管理权限，您可以创建用户定义的 AWS 资源标签并将其添加到 Storage Lens 组。您可以使用 AWS 资源标签根据部门、业务线或项目对资源进行分类。如果您有许多相同类型的资源，这样做会非常有用。通过应用标签，您可以根据分配给特定 Storage Lens 组的标签快速识别该组。

此外，当您向 Storage Lens 组添加 AWS 资源标签时，将激活基于属性的访问权限控制（ABAC）。ABAC 是一种基于属性（在本文中，也就是标签）定义权限的授权策略。您还可以在 IAM policy 中使用指定资源标签的条件，从而控制对 AWS 资源的访问。

主题

- 创建 Storage Lens 组
- 将 S3 Storage Lens 组附加到控制面板或从控制面板移除
- 可视化 Storage Lens 组数据
- 更新 Storage Lens 组
- 使用 Storage Lens 组管理 AWS 资源标签
- 列出所有 Storage Lens 组

- 查看 Storage Lens 组详细信息
- 删除 Storage Lens 组

创建 Storage Lens 组

以下示例演示如何使用 Amazon S3 控制台、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 创建 Amazon S3 Storage Lens 存储统计管理工具组。

使用 S3 控制台

创建 Storage Lens 组

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在页面顶部的导航栏中，选择当前所显示 AWS 区域的名称。接下来，选择要切换到的区域。
3. 在左侧导航窗格中，选择 Storage Lens 组。
4. 选择创建 Storage Lens 组。
5. 在常规下，查看您的主区域，然后输入 Storage Lens 组名称。
6. 在范围下，选择要应用于 Storage Lens 组的筛选条件。要应用多个筛选条件，请选择您的筛选条件，然后选择 AND 或 OR 逻辑运算符。
   - 对于前缀筛选条件，请选择前缀，然后输入前缀字符串。要添加多个前缀，请选择添加前缀。要移除前缀，请选择要移除的前缀旁的移除。
   - 对于对象标签筛选条件，请选择对象标签，然后输入对象的键值对。然后选择添加标签。要移除标签，请选择要移除的标签旁的移除。
   - 对于后缀筛选条件，请选择后缀，然后输入后缀字符串。要添加多个后缀，请选择添加后缀。要移除后缀，请选择要移除的后缀旁的移除。
   - 对于龄期筛选条件，请以天为单位指定对象的龄期范围。选择指定最小对象龄期，然后输入最小对象龄期。然后，选择指定最大对象龄期并输入最大对象龄期。
   - 对于大小筛选条件，请指定对象大小范围和衡量单位。选择指定最小对象大小，然后输入最小对象大小。选择指定最大对象大小，然后输入最大对象大小。
7. （可选）对于 AWS 资源标签，请添加键值对，然后选择添加标签。
8. 选择创建 Storage Lens 组。

使用 AWS CLI

以下示例 AWS CLI 命令将创建一个 Storage Lens 组。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

以下示例 AWS CLI 命令将创建一个具有两个 AWS 资源标签的 Storage Lens 组。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json \
--tags Key=k1,Value=v1 Key=k2,Value=v2
```

有关示例 JSON 配置，请参阅 Storage Lens 组配置。

使用适用于 Java 的 AWS SDK

以下 AWS SDK for Java 示例将创建一个 Storage Lens 组。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

Example – 创建包含单个筛选条件的 Storage Lens 组

以下示例将创建一个名为 *Marketing-Department* 的 Storage Lens 组。该组有一个对象龄期筛选条件，指定了 *30* 至 *90* 天的龄期范围。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;

public class CreateStorageLensGroupWithObjectAge {
    public static void main(String[] args) {
```

```
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupFilter objectAgeFilter = StorageLensGroupFilter.builder()
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90)
                            .build())
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(objectAgeFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example – 使用 **AND** 运算符创建包含多个筛选条件的 Storage Lens 组

以下示例将创建一个名为 *Marketing-Department* 的 Storage Lens 组。该组使用 AND 运算符来指示对象必须匹配所有筛选条件。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.S3Tag;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupAndOperator;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;


public class CreateStorageLensGroupWithAndFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create object tags.
            S3Tag tag1 = S3Tag.builder()
                    .key("object-tag-key-1")
                    .value("object-tag-value-1")
                    .build();
            S3Tag tag2 = S3Tag.builder()
                    .key("object-tag-key-2")
                    .value("object-tag-value-2")
                    .build();

            StorageLensGroupAndOperator andOperator =
 StorageLensGroupAndOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .matchAnyTag(tag1, tag2)
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90).build())
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L).build())
```

```
                    .build();

            StorageLensGroupFilter andFilter = StorageLensGroupFilter.builder()
                    .and(andOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(andFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example – 使用 **OR** 运算符创建包含多个筛选条件的 Storage Lens 组

以下示例将创建一个名为 *Marketing-Department* 的 Storage Lens 组。该组使用 OR 运算符来应用前缀筛选条件（*prefix-1*、*prefix-2*、*prefix3/sub-prefix-1*）或对象大小筛选条件（大小范围在 *1000* 字节至 *6000* 字节之间）。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
```

```
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupOrOperator;

public class CreateStorageLensGroupWithOrFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupOrOperator orOperator =
 StorageLensGroupOrOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L)
                            .build())
                    .build();

            StorageLensGroupFilter orFilter = StorageLensGroupFilter.builder()
                    .or(orOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(orFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example – 使用单个筛选条件和两个 AWS 资源标签创建 Storage Lens 组

以下示例将创建一个名为 *Marketing-Department* 的 Storage Lens 组，该组具有后缀筛选条件。此示例还向 Storage Lens 组添加了两个 AWS 资源标签。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.Tag;

public class CreateStorageLensGroupWithResourceTags {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create AWS resource tags.
            Tag resourceTag1 = Tag.builder()
                    .key("resource-tag-key-1")
                    .value("resource-tag-value-1")
                    .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
```

```
                    .value("resource-tag-value-2")
                    .build();

            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

有关示例 JSON 配置，请参阅 Storage Lens 组配置。

将 S3 Storage Lens 组附加到控制面板或从控制面板移除

在 Amazon S3 Storage Lens 存储统计管理工具中升级到高级级别后，您可以将 Storage Lens 组附加到控制面板。如果您有多个 Storage Lens 组，则可以根据需要包括或排除组。

您的 Storage Lens 组必须位于控制面板账户中指定的主区域内。将 Storage Lens 组附加到控制面板后，您将在 48 小时内在指标导出中收到新增的 Storage Lens 组聚合数据。

> **Note**
>
> 如果要查看 Storage Lens 组的聚合指标，则必须将其附加到 Storage Lens 控制面板。有关 Storage Lens 组 JSON 配置文件的示例，请参阅 JSON 格式的 S3 Storage Lens 存储统计管理工具示例配置（带 Storage Lens 组）。

将 Storage Lens 组附加到 S3 Storage Lens 存储统计管理工具控制面板

将 Storage Lens 组附加到 Storage Lens 控制面板

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，在 Storage Lens 下选择控制面板。
3. 选择要将 Storage Lens 组附加到的 Storage Lens 控制面板的选项按钮。
4. 选择编辑。
5. 在 Metrics selection（指标选择）下方，请选择 Advanced metrics and recommendations（高级指标和建议）。
6. 选择 Storage Lens 组聚合。

   > **Note**
   >
   > 默认情况下，还会选中高级指标。但是，您也可以取消选中此设置，因为聚合 Storage Lens 组数据并非必须。

7. 向下滚动到 Storage Lens 组聚合，然后指定要在数据聚合中包括或排除的一个或多个 Storage Lens 组。您可以使用以下筛选条件选项：

   - 如果要包括某些 Storage Lens 组，请选择包括 Storage Lens 组。在要包括的 Storage Lens 组下，选择您的 Storage Lens 组。
   - 如果要包括所有 Storage Lens 组，请选择包括此账户中主区域中的所有 Storage Lens 组。
   - 如果要排除某些 Storage Lens 组，请选择排除 Storage Lens 组。在要排除的 Storage Lens 组下，选择要排除的 Storage Lens 组。

8. 选择 Save changes（保存更改）。如果您正确配置了 Storage Lens 组，则将在 48 小时内在控制面板中看到新增的 Storage Lens 组聚合数据。

从 S3 Storage Lens 存储统计管理工具控制面板移除 Storage Lens 组

从 S3 Storage Lens 存储统计管理工具控制面板移除 Storage Lens 组

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，在 Storage Lens 下选择控制面板。
3. 选择要从中移除 Storage Lens 组的 Storage Lens 控制面板的选项按钮。
4. 选择查看控制面板配置。
5. 选择编辑。
6. 向下滚动到指标选择部分。
7. 在 Storage Lens 组聚合下，选择要移除的 Storage Lens 组旁的 X。这将移除 Storage Lens 组。

   如果您在控制面板中包括了所有 Storage Lens 组，请清除包括此账户中主区域中的所有 Storage Lens 组旁的复选框。
8. 选择 Save changes（保存更改）。

> **Note**
>
> 最多需要等待 48 小时，控制面板才会反映配置更新。

可视化 Storage Lens 组数据

通过将 Storage Lens 组附加到 Amazon S3 Storage Lens 存储统计管理工具控制面板，您可以将该组数据可视化。在控制面板配置中将 Storage Lens 组包含在 Storage Lens 组聚合中后，最长可能需要 48 小时才能在控制面板中显示 Storage Lens 组的数据。

更新控制面板配置后，任何新附加的 Storage Lens 组都将显示在 Storage Lens 组选项卡下的可用资源列表中。您还可以在概览选项卡中，通过按其它维度对数据进行切片，来进一步分析存储使用情况。例如，您可以选择前 3 个类别下列出的项目之一，然后选择分析依据，按另一个维度对数据进行切片。不能应用与筛选条件本身相同的维度。

> **Note**
>
> 您不能将 Storage Lens 组筛选条件与前缀筛选条件一起应用，反之亦然。您也无法使用前缀筛选条件进一步分析 Storage Lens 组。

您可以使用 Amazon S3 Storage Lens 存储统计管理工具控制面板中的 Storage Lens 组选项卡，为附加到控制面板的 Storage Lens 组自定义数据可视化。您可以可视化附加到控制面板的某些 Storage Lens 组的数据，也可以将所有组的数据可视化。

在 S3 Storage Lens 存储统计管理工具控制面板中可视化 Storage Lens 组数据时，请注意以下几点：

- S3 Storage Lens 存储统计管理工具聚合了所有匹配的 Storage Lens 组下某个对象的使用情况指标。因此，如果某个对象与两个或更多 Storage Lens 组的筛选条件相匹配，则在整个存储使用情况中，您会看到同一对象的重复计数。
- 对象必须与您在 Storage Lens 组中包含的筛选条件相匹配。如果没有对象与您在 Storage Lens 组中包含的筛选条件相匹配，则不会生成任何指标。要确定是否有任何未分配的对象，请在控制面板中查看账户级别和桶级别的对象总数。

更新 Storage Lens 组

以下示例演示如何更新 Amazon S3 Storage Lens 存储统计管理工具组。您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK for Java 更新 Storage Lens 组。

使用 S3 控制台

更新 Storage Lens 组

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 组。
3. 在 Storage Lens 组下，选择要更新的 Storage Lens 组。
4. 在范围下，选择编辑。
5. 在范围页面上，选择要应用于 Storage Lens 组的筛选条件。要应用多个筛选条件，请选择您的筛选条件，然后选择 AND 或 OR 逻辑运算符。
   - 对于前缀筛选条件，请选择前缀，然后输入前缀字符串。要添加多个前缀，请选择添加前缀。要移除前缀，请选择要移除的前缀旁的移除。
   - 对于对象标签筛选条件，请输入对象的键值对。然后选择添加标签。要移除标签，请选择要移除的标签旁的移除。
   - 对于后缀筛选条件，请选择后缀，然后输入后缀字符串。要添加多个后缀，请选择添加后缀。要移除后缀，请选择要移除的后缀旁的移除。

- 对于龄期筛选条件，请以天为单位指定对象的龄期范围。选择指定最小对象龄期，然后输入最小对象龄期。对于指定最大对象龄期，请输入最大对象龄期。
- 对于大小筛选条件，请指定对象大小范围和衡量单位。选择指定最小对象大小，然后输入最小对象大小。对于指定最大对象大小，请输入最大对象大小。

6. 选择 Save changes（保存更改）。将显示 Storage Lens 组的详细信息页面。
7. （可选）如果要添加新的 AWS 资源标签，请滚动到 AWS 资源标签部分，然后选择添加标签。此时将显示 Add tags (添加标签) 页面。

   添加新的键值对，然后选择保存更改。将显示 Storage Lens 组的详细信息页面。
8. （可选）如果要移除现有 AWS 资源标签，请滚动到 AWS 资源标签部分，然后选择资源标签。然后选择删除。将出现删除 AWS 标签对话框。

   再次选择删除可永久删除 AWS 资源标签。

   > **Note**
   >
   > 永久删除 AWS 资源标签后，该标签将无法还原。

使用 AWS CLI

以下 AWS CLI 示例命令将返回名为 *marketing-department* 的 Storage Lens 组的配置详细信息。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

以下 AWS CLI 示例将更新 Storage Lens 组。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control update-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

有关示例 JSON 配置，请参阅 Storage Lens 组配置。

使用适用于 Java 的 AWS SDK

以下 AWS SDK for Java 示例将返回账户 *111122223333* 中 *Marketing-Department* Storage Lens 组的配置详细信息。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

以下示例将更新账户 *111122223333* 中名为 *Marketing-Department* 的 Storage Lens 组。此示例将更新控制面板范围，使其包含与以下任何后缀匹配的对象：*.png*、*.gif*、*.jpg* 或 *.jpeg*。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.UpdateStorageLensGroupRequest;

public class UpdateStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create updated filter.
            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg", ".jpeg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            UpdateStorageLensGroupRequest updateStorageLensGroupRequest =
 UpdateStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId)
                    .build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.updateStorageLensGroup(updateStorageLensGroupRequest);
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

有关示例 JSON 配置，请参阅 Storage Lens 组配置。

使用 Storage Lens 组管理 AWS 资源标签

每个 Amazon S3 Storage Lens 存储统计管理工具组都视为一个 AWS 资源，具有自己的 Amazon 资源名称（ARN）。因此，在配置 Storage Lens 组时，可以选择向该组添加 AWS 资源标签。最多可以为每个 Storage Lens 组添加 50 个标签。要创建带标签的 Storage Lens 组，您必须拥有 `s3:CreateStorageLensGroup` 和 `s3:TagResource` 权限。

您可以使用 AWS 资源标签根据部门、业务线或项目对资源进行分类。如果您有许多相同类型的资源，这样做会非常有用。通过应用标签，您可以根据分配给特定 Storage Lens 组的标签快速识别该组。您还可以使用标签对成本进行跟踪和分配。

此外，当您向 Storage Lens 组添加 AWS 资源标签时，将激活基于属性的访问权限控制（ABAC）。ABAC 是一种基于属性（在本文中，也就是标签）定义权限的授权策略。您还可以在 IAM policy 中使用指定资源标签的条件，从而控制对 AWS 资源的访问。

您可以修改标签的密钥和值，还可以随时删除资源的标签。此外，请注意以下限制：

- 标签键和标签值区分大小写。
- 如果您添加的标签的值与该实例上现有标签的值相同，新的值就会覆盖旧值。
- 如果删除资源，资源的所有标签也会被删除。
- 不要在 AWS 资源标签中包含隐私或敏感数据。
- 不支持系统标签（标签键以 `aws:` 开头）。
- 每个标签键的长度不能超过 128 个字符。每个标签值的长度不能超过 256 个字符。

以下示例演示如何将 AWS 资源标签与 Storage Lens 组配合使用。

主题

向 Storage Lens 组添加 AWS 资源标签

以下示例演示如何将 AWS 资源标签添加到 Amazon S3 Storage Lens 存储统计管理工具组。您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK for Java 添加资源标签。

使用 S3 控制台

向 Storage Lens 组添加 AWS 资源标签

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 组。
3. 在 Storage Lens 组下，选择要更新的 Storage Lens 组。
4. 在 AWS 资源标签下，选择添加标签。
5. 在添加标签页面上，添加新的键值对。

   > **Note**
   >
   > 如果添加的新标签与现有标签具有相同的键，则将覆盖之前的标签值。

6. （可选）要添加多个新标签，请再次选择添加标签以继续添加新条目。您最多可以将 50 个 AWS 资源标签添加到 Storage Lens 组。
7. （可选）如果要移除新添加的条目，请选择要移除的标签旁的移除。
8. 选择 Save changes（保存更改）。

使用 AWS CLI

以下示例 AWS CLI 命令将向名为 *marketing-department* 的现有 Storage Lens 组添加两个资源标签。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v1 Key=k2,Value=v2
```

使用适用于 Java 的 AWS SDK

以下 AWS SDK for Java 示例将向现有 Storage Lens 组添加两个 AWS 资源标签。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class TagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag resourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-value-1")
                .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
```

```
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

更新 Storage Lens 组标签值

以下示例演示如何使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK for Java 更新 Storage Lens 组标签值。

使用 S3 控制台

更新 Storage Lens 组的 AWS 资源标签

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 组。
3. 在 Storage Lens 组下，选择要更新的 Storage Lens 组。
4. 在 AWS 资源标签下，选择要更新的标签。
5. 使用与要更新的键值对相同的键添加新标签值。选择对勾图标以更新标签值。

   > **Note**
   >
   > 如果添加的新标签与现有标签具有相同的键，则将覆盖之前的标签值。

6. （可选）如果要添加新标签，请选择添加标签以添加新条目。此时将显示 Add tags (添加标签) 页面。

   最多可以为 Storage Lens 组添加 50 个 AWS 资源标签。添加完新标签后，选择保存更改。
7. （可选）如果要移除新添加的条目，请选择要移除的标签旁的移除。完成移除标签后，选择保存更改。

使用 AWS CLI

以下示例 AWS CLI 命令将更新名为 *marketing-department* 的 Storage Lens 组的两个标签值。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v3 Key=k2,Value=v4
```

使用适用于 Java 的 AWS SDK

以下 AWS SDK for Java 示例将更新 Storage Lens 组的两个标签值。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class UpdateTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag updatedResourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-updated-value-1")
                .build();
            Tag updatedResourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-updated-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(updatedResourceTag1, updatedResourceTag2)
                    .accountId(accountId)
```

```
                .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                .region(Region.US_WEST_2)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

从 Storage Lens 组删除 AWS 资源标签

以下示例演示如何从 Storage Lens 组删除 AWS 资源标签。您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK for Java 删除标签。

使用 S3 控制台

从 Storage Lens 组删除 AWS 资源标签

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 组。
3. 在 Storage Lens 组下，选择要更新的 Storage Lens 组。
4. 在 AWS 资源标签下，选择要删除的键值对。
5. 选择 Delete (删除)。将出现删除 AWS 资源标签对话框。

> **Note**
>
> 如果使用标签控制访问，则继续执行此操作可能会影响相关资源。永久删除标签后，该标签将无法还原。

6. 选择删除，永久性删除该键值对。

使用 AWS CLI

以下 AWS CLI 命令将从现有 Storage Lens 组中删除两个 AWS 资源标签：要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control untag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/Marketing-
Department \
--region us-east-1 --tag-keys k1 k2
```

使用适用于 Java 的 AWS SDK

以下 AWS SDK for Java 示例将从账户 *111122223333* 中具有您所指定的 Amazon 资源名称（ARN）的 Storage Lens 组中，删除两个 AWS 资源标签。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.UntagResourceRequest;

public class UntagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            String tagKey1 = "resource-tag-key-1";
            String tagKey2 = "resource-tag-key-2";
            UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tagKeys(tagKey1, tagKey2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
```

```
            s3ControlClient.untagResource(untagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

列出 Storage Lens 组标签

以下示例演示如何列出与 Storage Lens 组关联的 AWS 资源标签。您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK for Java 列出标签。

使用 S3 控制台

查看 Storage Lens 组的标签和标签值列表

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 组。
3. 在 Storage Lens 组下，选择您感兴趣的 Storage Lens 组。
4. 向下滚动到 AWS 资源标签部分。所有添加到您的 Storage Lens 组的用户定义的 AWS 资源标签及其标签值都将列出。

使用 AWS CLI

以下 AWS CLI 示例命令将列出名为 *marketing-department* 的 Storage Lens 组的所有 Storage Lens 组标签值。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control list-tags-for-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1
```

使用适用于 Java 的 AWS SDK

以下 AWS SDK for Java 示例将为您指定的 Storage Lens 组 Amazon 资源名称（ARN）列出 Storage Lens 组标签值。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListTagsForResourceResponse response =
 s3ControlClient.listTagsForResource(listTagsForResourceRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
```

```
}
```

列出所有 Storage Lens 组

以下示例演示如何列出 AWS 账户和主区域中的所有 Amazon S3 Storage Lens 存储统计管理工具组。这些示例显示了如何使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK for Java 列出所有 Storage Lens 组。

使用 S3 控制台

列出账户和主区域中的所有 Storage Lens 组

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 组。
3. 在 Storage Lens 组下，将显示您账户中的 Storage Lens 组列表。

使用 AWS CLI

以下 AWS CLI 示例将列出您账户的所有 Storage Lens 组。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control list-storage-lens-groups --account-id 111122223333 \
--region us-east-1
```

使用适用于 Java 的 AWS SDK

以下 AWS SDK for Java 示例将列出账户 *111122223333* 的 Storage Lens 组。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsRequest;
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsResponse;
```

```
public class ListStorageLensGroups {
    public static void main(String[] args) {
        String accountId = "111122223333";

        try {
            ListStorageLensGroupsRequest listStorageLensGroupsRequest =
 ListStorageLensGroupsRequest.builder()
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListStorageLensGroupsResponse response =
 s3ControlClient.listStorageLensGroups(listStorageLensGroupsRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

查看 Storage Lens 组详细信息

以下示例演示如何查看 Amazon S3 Storage Lens 存储统计管理工具组配置详细信息。您可以使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK for Java 查看这些详细信息。

使用 S3 控制台

查看 Storage Lens 组配置详细信息

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 组。
3. 在 Storage Lens 组下，选择您感兴趣的 Storage Lens 组旁的选项按钮。

4. 请选择 View details (查看详细信息)。现在，您可以查看 Storage Lens 组的详细信息。

使用 AWS CLI

以下 AWS CLI 示例将返回 Storage Lens 组的配置详细信息。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

使用适用于 Java 的 AWS SDK

以下 AWS SDK for Java 示例将返回账户 *111122223333* 中名为 *Marketing-Department* 的 Storage Lens 组的配置详细信息。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
```

```
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

删除 Storage Lens 组

以下示例演示如何使用 Amazon S3 控制台、AWS Command Line Interface（AWS CLI）和 AWS SDK for Java 删除 Amazon S3 Storage Lens 存储统计管理工具组。

使用 S3 控制台

删除 Storage Lens 组

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Storage Lens 组。
3. 在 Storage Lens 组下，选择要删除的 Storage Lens 组旁的选项按钮。
4. 选择 Delete (删除)。将显示删除 Storage Lens 组对话框。
5. 再次选择删除即可永久删除 Storage Lens 组。

> **Note**
>
> Storage Lens 组在删除后无法还原。

使用 AWS CLI

以下 AWS CLI 示例将删除名为 *marketing-department* 的 Storage Lens 组。要使用此示例命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control delete-storage-lens-group --account-id 111122223333 \
```

```
--region us-east-1 --name marketing-department
```

使用适用于 Java 的 AWS SDK

以下 AWS SDK for Java 示例将删除账户 *111122223333* 中名为 *Marketing-Department* 的 Storage Lens 组。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.DeleteStorageLensGroupRequest;

public class DeleteStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            DeleteStorageLensGroupRequest deleteStorageLensGroupRequest =
 DeleteStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.deleteStorageLensGroup(deleteStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 跟踪 Amazon S3 请求AWS X-Ray

AWS X-Ray 收集有关应用程序所处理请求的数据。然后，您可以查看和筛选数据，以识别和排查分布式应用程序和微服务架构中的性能问题和错误。对于任何被跟踪的对您应用程序的请求，它将显示请求和响应的详细信息，以及您的应用程序对下游 AWS 资源、微服务、数据库和 HTTP Web API 进行的调用的详细信息。

有关更多信息，请参阅 AWS X-Ray 开发人员指南中的什么是 AWS X-Ray？

主题

X-Ray 如何与 Amazon S3 配合使用

AWS X-Ray 支持 Amazon S3 的跟踪上下文传播，因此您可以在端到端请求通过整个应用程序时查看这些请求。X-Ray 聚合了 Amazon S3、AWS Lambda 和 Amazon EC2 等单个服务生成的数据，以及构成应用程序的许多资源。它为您提供了应用程序性能的总体视图。

Amazon S3 与 X-Ray 集成，以传播跟踪上下文，并为您提供一个带有上游和下游节点的请求链。如果上游服务在其 S3 请求中包含有效格式的跟踪标头，则 Amazon S3 会在将事件通知发送到下游服务（如 Lambda、Amazon SQS 和 Amazon SNS）时传递跟踪标头。如果您将所有这些服务与 X-Ray 主动集成，它们将链接在一个请求链中，以便为您提供 Amazon S3 请求的完整详细信息。

要通过 Amazon S3 发送 X-Ray 跟踪标头，您必须在请求中包含格式化的 X-Amzn-Trace-ID。您还可以使用 AWS X-Ray 开发工具包对 Amazon S3 客户端进行检测。有关受支持开发工具包的列表，请参阅 AWS X-Ray 文档。

Service Map

X-Ray 服务地图可以近实时地向您展示 Amazon S3 与应用程序中其他 AWS 服务和资源之间的关系。要使用 X-Ray Service Map 查看端到端请求，您可以使用 X-Ray 控制台查看 Amazon S3 与应用程序使用的其他服务之间的连接映射。您可以轻松检测出出现高延迟的位置，并可视化这些服务的节点分布情况，进而确定影响应用程序性能的具体服务和路径。

X-Ray Analytics

您还可以使用 X-Ray Analytics 控制台来分析跟踪、查看延迟和故障率等指标并生成帮助您识别和排查问题的见解。此控制台还会显示平均延迟和失败率等指标。有关更多信息，请参阅 AWS X-Ray 开发人员指南中的 AWS X-Ray 控制台。

可用区

AWS X-Ray 对 Amazon S3 的支持可用于所有 AWS X-Ray 区域。有关更多信息，请参阅 AWS X-Ray 开发人员指南中的 Amazon S3 和 AWS X-Ray。

使用 Amazon S3 托管静态网站

您可以使用 Amazon S3 托管静态网站。在静态网站上，单独的网页包含静态内容。它们也可能包含客户端脚本。

通过对比得知，动态网站依赖服务器端处理，包括诸如 PHP、JSP 或 ASP.NET 的服务器端脚本。Amazon S3 不支持服务器端脚本编写，但 AWS 具有其他用于托管动态网站的资源。要了解有关 AWS 上的网站托管的更多信息，请参阅 Web 托管。

> **Note**
>
> 可以使用 AWS Amplify 控制台托管单页 Web 应用程序。AWS Amplify 控制台支持使用单页应用程序框架（例如，React JS、Vue JS、Angular JS 和 Nuxt）和静态站点生成器（例如，Gatsby JS、React-static、Jekyll 和 Hugo）构建的单页应用程序。有关更多信息，请参阅《AWS Amplify 控制台用户指南》中的入门。
>
> Amazon S3 网站端点不支持 HTTPS。如果要使用 HTTPS，则可以使用 Amazon CloudFront 为 Amazon S3 上托管的静态网站提供服务。有关更多信息，请参阅如何使用 CloudFront 为我的 Amazon S3 存储桶提供 HTTPS 请求？ 要使用自定义域的 HTTPS，请参阅使用注册到 Route 53 的自定义域配置静态网站。

有关在 Amazon S3 上托管静态网站的更多信息（包括说明和分步演练），请参阅以下主题。

主题

网站端点

当您将存储桶配置为静态网站时，该网站在存储桶的 AWS 区域 特定的网站端点上可用。网站端点不同于您在其上发送 REST API 请求的端点。有关端点之间的差异的更多信息，请参阅 网站端点和 REST API 端点之间的主要区别。

根据您所在的区域，Amazon S3 网站端点采用以下两种格式之一。

- s3-website 短横线 (-) 区域 - `http://bucket-name.s3-website-Region.amazonaws.com`
- s3-网站 点 (.) 区域 - `http://bucket-name.s3-website.Region.amazonaws.com`

此 URL 将返回您为该网站配置的默认索引文档。有关 Amazon S3 网站端点的完整列表，请参阅 Amazon S3 网站端点。

> **Note**
>
> 为了增强 Amazon S3 静态网站的安全性，Amazon S3 网站端点域（例如 s3-website-us-east-1.amazonaws.com 或 s3-website.ap-south-1.amazonaws.com）已在 Public Suffix List (PSL)中注册。为进一步增强安全性，如果您需要在 Amazon S3 静态网站的域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造 (CSRF) 攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

如果您希望您的网站公开，您必须使您的所有内容对您的客户公开可读，以便您的客户可以在网站端点访问这些内容。有关更多信息，请参阅 设置访问网站的权限。

> **Important**
>
> Amazon S3 网站端点不支持 HTTPS 或接入点。如果要使用 HTTPS，则可以使用 Amazon CloudFront 为 Amazon S3 上托管的静态网站提供服务。有关更多信息，请参阅如何使用 CloudFront 为我的 Amazon S3 存储桶提供 HTTPS 请求？ 要使用自定义域的 HTTPS，请参阅使用注册到 Route 53 的自定义域配置静态网站。
>
> 申请方付款存储桶不允许通过网站端点进行访问。对此类存储桶的任何请求都会收到 403 拒绝访问响应。有关更多信息，请参阅 使用申请方付款存储桶进行存储传输和使用。

主题

网站端点示例

以下示例演示如何访问配置为静态网站的 Amazon S3 存储桶。

Example – 请求根级别的对象

要请求存储在存储桶根级别的特定对象，请使用以下 URL 结构。

```
http://bucket-name.s3-website.Region.amazonaws.com/object-name
```

例如，以下 URL 请求存储在存储桶中根级别的 photo.jpg 对象。

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/photo.jpg
```

Example – 请求前缀中的对象

要请求存储在存储桶的文件夹中的对象，请使用此 URL 结构。

```
http://bucket-name.s3-website.Region.amazonaws.com/folder-name/object-name
```

以下 URL 请求存储桶中的 docs/doc1.html 对象。

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/docs/doc1.html
```

添加 DNS 别名记录

如果您拥有已注册的域，则可以添加指向 Amazon S3 网站端点的 DNS CNAME 条目。例如，如果您注册了 www.example-bucket.com 域，则可以创建存储桶 www.example-bucket.com，并添加指向 www.example-bucket.com.s3-website.*Region*.amazonaws.com 的 DNS CNAME 记录。对 http://www.example-bucket.com 的所有请求都将路由到 www.example-bucket.com.s3-website.*Region*.amazonaws.com。

有关更多信息，请参阅 使用 CNAME 记录自定义 Amazon S3 URL。

将自定义域与 Route 53 结合使用

您可以使用自己的向 Amazon Route 53 注册的域（例如，`example.com`）来提供您的内容，而不是使用 Amazon S3 网站端点访问网站。您可以将 Amazon S3 与 Route 53 结合使用以在根域中托管网站。例如，如果您拥有根域 `example.com` 且在 Amazon S3 上托管您的网站，则您的网站访问者可以通过输入 `http://www.example.com` 或 `http://example.com` 从他们的浏览器访问该站点。

有关示例演练的信息，请参阅 教程：使用注册到 Route 53 的自定义域配置静态网站。

网站端点和 REST API 端点之间的主要区别

Amazon S3 网站端点针对通过 Web 浏览器访问进行了优化。下表总结 REST API 端点和网站端点之间的主要区别。

主要区别	REST API 端点	网站端点
访问控制	同时支持公有和私有内容	仅支持公开可读的内容
错误消息处理	返回 XML 格式的错误响应	返回 HTML 文档
重定向支持	不适用	同时支持对象级和存储桶级重定向
支持的请求	支持所有存储桶和对象操作	仅支持对象上的 GET 和 HEAD 请求
对存储桶根级的 GET 和 HEAD 请求的响应	返回存储桶中对象键的列表	返回在网站配置中指定的索引文档
安全套接字层 (SSL) 支持	支持 SSL 连接	不支持 SSL 连接

有关 Amazon S3 端点的完整列表，请参阅《AWS 一般参考》中的 Amazon S3 端点和限额。

启用网站托管

将存储桶配置为静态网站时，您必须启用静态网站托管、配置索引文档和设置权限。

您可以使用 Amazon S3 控制台、REST API、AWS SDK、AWS CLI 或 AWS CloudFormation 启用静态网站托管。

要使用自定义域配置您的网站，请参阅 教程：使用注册到 Route 53 的自定义域配置静态网站。

使用 S3 控制台

启用静态网站托管

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，请选择要为其启用静态网站托管的存储桶的名称。
3. 选择属性。
4. 在静态网站托管下，请选择编辑。
5. 请选择使用此存储桶托管网站。
6. 在静态网站托管下，请选择启用。
7. 在 Index document (索引文档) 中，输入索引文档的文件名，通常为 `index.html`。

   索引文档名称区分大小写，并且必须与您计划上传到 S3 存储桶的 HTML 索引文档的文件名完全匹配。当您为网站托管配置存储桶时，您必须指定索引文档。当对根域或任何子文件夹发出请求时，Amazon S3 将返回此索引文档。有关更多信息，请参阅 配置索引文档。
8. 要为 4XX 类错误提供您自己的自定义错误文档，请在错误文档中输入自定义错误文档文件名。

   错误文档名称区分大小写，并且必须与您计划上传到 S3 存储桶的 HTML 错误文档的文件名完全匹配。如果未指定自定义错误文档并发生错误，Amazon S3 返回默认 HTML 错误文档。有关更多信息，请参阅 配置自定义错误文档。
9. （可选）如果要指定高级重定向规则，请在 Redirection rules（重定向规则）中，输入 JSON 来描述规则。

   例如，您可以根据请求中的特定对象键名或前缀按条件路由请求。有关更多信息，请参阅 配置重新导向规则以使用高级条件重新导向。
10. 选择 Save changes（保存更改）。

    Amazon S3 为您的存储桶启用静态网站托管。在页面底部的静态网站托管下，您可以看到存储桶的网站端点。
11. 在静态网站托管下，记下端点。

端点是存储桶的 Amazon S3 网站端点。将存储桶配置为静态网站后，您可以使用此端点来测试您的网站。

使用 REST API

有关直接发送 REST 请求以启用静态网站托管的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的以下部分：

- PUT Bucket website
- GET Bucket website
- DELETE Bucket website

使用 AWS SDK

若要在 Amazon S3 上托管静态网站，需要为网站托管配置 Amazon S3 存储桶，然后将您的网站内容上传到该存储桶。您还可以使用 AWS SDK 以编程方式创建、更新和删除网站配置。SDK 围绕 Amazon S3 REST API 提供封装类。如果应用程序需要它，可以直接从应用程序发送 REST API 请求。

.NET

以下示例说明如何使用 AWS SDK for .NET 管理存储桶的网站配置。要将网站配置添加到存储桶，请提供存储桶名称和网站配置。网站配置必须包含索引文档，并且可包含可选的错误文档。这些文档必须存储在存储桶中。有关更多信息，请参阅 PUT Bucket 网站。有关 Amazon S3 网站功能的更多信息，请参阅 使用 Amazon S3 托管静态网站。

以下 C# 代码示例将网站配置添加到指定的存储桶。该配置指定索引文档和错误文档名称。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class WebsiteConfigTest
```

```
   {
        private const string bucketName = "*** bucket name ***";
        private const string indexDocumentSuffix = "*** index object key ***"; //
For example, index.html.
        private const string errorDocument = "*** error object key ***"; // For
example, error.html.
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddWebsiteConfigurationAsync(bucketName, indexDocumentSuffix,
errorDocument).Wait();
        }

        static async Task AddWebsiteConfigurationAsync(string bucketName,
                                            string indexDocumentSuffix,
                                            string errorDocument)
        {
            try
            {
                // 1. Put the website configuration.
                PutBucketWebsiteRequest putRequest = new PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument
                    }
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);

                // 2. Get the website configuration.
                GetBucketWebsiteRequest getRequest = new GetBucketWebsiteRequest()
                {
                    BucketName = bucketName
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
```

```
                Console.WriteLine("Index document: {0}",
 getResponse.WebsiteConfiguration.IndexDocumentSuffix);
                Console.WriteLine("Error document: {0}",
 getResponse.WebsiteConfiguration.ErrorDocument);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

以下 PHP 示例将网站配置添加到指定的存储桶。create_website_config 方法显式提供索引文档和错误文档名称。该示例还检索网站配置并输出响应。有关 Amazon S3 网站功能的更多信息，请参阅 使用 Amazon S3 托管静态网站。

有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);


// Add the website configuration.
$s3->putBucketWebsite([
    'Bucket'                => $bucket,
```

```
    'WebsiteConfiguration'  => [
        'IndexDocument' => ['Suffix' => 'index.html'],
        'ErrorDocument' => ['Key' => 'error.html']
    ]
]);

// Retrieve the website configuration.
$result = $s3->getBucketWebsite([
    'Bucket' => $bucket
]);
echo $result->getPath('IndexDocument/Suffix');

// Delete the website configuration.
$s3->deleteBucketWebsite([
    'Bucket' => $bucket
]);
```

使用 AWS CLI

有关使用 AWS CLI 将 S3 存储桶配置为静态网站的更多信息，请参阅《AWS CLI 命令参考》中的网站。

接下来，您必须配置索引文档并设置权限。有关信息，请参阅 配置索引文档 和 设置访问网站的权限。

还可以有选择性地配置错误文档、Web 流量日志记录或重定向。

配置索引文档

启用网站托管时，还必须配置和上传索引文档。索引文档 是在对网站的根或任何子文件夹发出请求时 Amazon S3 返回的网页。例如，如果用户在浏览器中输入 `http://www.example.com`，则该用户没有请求任何特定页面。在这种情况下，Amazon S3 将提供索引文档，该文档有时也可称为默认页面。

当您为存储桶启用静态网站托管时，您可以输入索引文档的名称（例如，`index.html`）。为存储桶启用静态网站托管后，您可以将具有索引文档名称的 HTML 文件上传到存储桶。

根级 URL 的尾部斜杠是可选的。例如，如果您将具有 `index.html` 的网站配置为索引文档，以下任意一个 URL 将返回 `index.html`。

```
http://example-bucket.s3-website.Region.amazonaws.com/
```

```
http://example-bucket.s3-website.Region.amazonaws.com
```

有关 Amazon S3 网站端点的更多信息，请参阅 网站端点。

索引文档和文件夹

在 Amazon S3 中，存储桶是对象的平面容器。它不会像计算机上的文件系统那样提供任何分层组织。但是，您可以通过使用表示文件夹结构的对象键名创建逻辑层级结构。

例如，考虑具有三个对象（具有以下键名）的存储桶。虽然它们没有按任何物理分层组织进行存储，但您可以从键名推断以下逻辑文件夹结构：

- sample1.jpg — 对象位于存储桶的根级。
- photos/2006/Jan/sample2.jpg — 对象位于 photos/2006/Jan 子文件夹中。
- photos/2006/Feb/sample3.jpg — 对象位于 photos/2006/Feb 子文件夹中。

在 Amazon S3 控制台中，您还可以在存储桶中创建文件夹。例如，您可以创建名为 photos 的文件夹。您可以将对象上传到存储桶或该存储桶中的 photos 文件夹。如果您将对象 sample.jpg 添加到存储桶，则键名为 sample.jpg。如果您将对象上传到 photos 文件夹，则对象键名为 photos/sample.jpg。

如果您在存储桶中创建了文件夹结构，则您必须在每个级别上都具有索引文档。在每个文件夹中，索引文档必须具有相同的名称，例如，index.html。当用户指定类似于文件夹查找的 URL 时，是否存在尾部斜杠将决定网站的行为。例如，以下具有尾部反斜杠的 URL 将返回 photos/index.html 索引文档。

```
http://bucket-name.s3-website.Region.amazonaws.com/photos/
```

但是，如果您从之前的 URL 排除了尾部斜杠，则 Amazon S3 将首先在存储段中查找对象 photos。如果未找到 photos 对象，它将搜索索引文档 photos/index.html。如果找到该文档，则 Amazon S3 将返回 302 Found 消息并指向 photos/ 密钥。对于对 photos/ 的后续请求，Amazon S3 将返回 photos/index.html。如果未找到索引文档，Amazon S3 将返回错误。

配置索引文档

要使用 S3 控制台配置索引文档，请使用以下步骤。您还可以使用 REST API、AWS SDK、AWS CLI 或 AWS CloudFormation 配置索引文档。

> **Note**
>
> 在启用版本控制的存储桶中，您可以上传 index.html 的多个副本，但只解析为最新的版本。有关使用 S3 版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。

当您为存储桶启用静态网站托管时，您可以输入索引文档的名称（例如，**index.html**）。为存储桶启用静态网站托管后，您可以将具有此索引文档名称的 HTML 文件上传到存储桶。

配置索引文档

1. 创建 index.html 文件。

   如果您没有 index.html 文件，则可以使用以下 HTML 创建一个：

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. 将索引文件保存在本地。

   索引文档文件名必须与您在静态网站托管对话框中输入的索引文档名称完全匹配。索引文档名称区分大小写。例如，如果在静态网站托管对话框中为索引文档名称输入 index.html，则索引文档文件名也必须是 index.html，而不是 Index.html。

3. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
4. 在存储桶列表中，请选择要用于托管静态网站的存储桶的名称。
5. 为您的存储桶启用静态网站托管，并输入索引文档的确切名称（例如 index.html）。有关更多信息，请参阅 启用网站托管。

   启用静态网站托管后，继续执行步骤 6。

6. 要将索引文档上传到存储桶，请执行以下操作之一：

   - 将索引文件拖放到控制台存储桶列表中。

- 选择上传，然后按照提示选择并上传索引文件。

  如需分步指导，请参阅 上传对象。

7. （可选）将其他网站内容上传到您的存储桶。

接下来，您必须设置用于访问网站的权限。有关信息，请参阅 设置访问网站的权限。

还可以有选择性地配置错误文档、Web 流量日志记录或重定向。

配置自定义错误文档

将存储桶配置为静态网站后，当出现错误时，Amazon S3 返回 HTML 错误文档。您可以选择使用自定义错误文档配置存储桶，以便在发生错误时 Amazon S3 返回该文档。

> **Note**
>
> 当出现错误时，某些浏览器将显示自己的错误消息，而忽略 Amazon S3 返回的错误文档。例如，当出现 HTTP 404 Not Found (HTTP 404 未找到) 错误时，Google Chrome 可能忽略 Amazon S3 返回的错误文档并显示自己的错误。

主题

Amazon S3 HTTP 响应代码

下表列出了出现错误时 Amazon S3 返回的 HTTP 响应代码的子集。

HTTP 错误代码	描述
301 永久移动	当用户将请求直接发送到 Amazon S3 网站端点（`http://s3-website.Region.amazonaws.com/`）时，Amazon S3 将返回 301 Moved Permanently（301 永久移动）响应并将这些请求重定向到 `https://aws.amazon.com/s3/`。

HTTP 错误代码	描述
302 Found (302 已找到)	当 Amazon S3 接收对不包含尾部斜杠的键 x (`http://bucket-name .s3-website. Region.amazonaws.com/x `) 的请求时，它首先查找键名为 x 的对象。如果未找到对象，则 Amazon S3 确定该请求是针对子文件夹 x 发出的，并通过在末尾添加斜杠重定向请求并返回 302 Found (302 已找到)。
304 Not Modified (304 未修改)	Amazon S3 使用请求标头 `If-Modified-Since` 、`If-Unmodified-Since` 、`If-Match` 和/或 `If-None-Match` 来确定请求的对象与客户端具有的缓存副本是否相同。如果对象相同，网站端点将返回 304 Not Modified 响应。
400 Malformed Request	当用户尝试通过错误的地区端点访问存储桶时，网站端点的响应包含 400 Malformed Request。
403 禁止访问	当用户请求转换为不可公开读取的对象时，网站端点的响应包含 403 禁止访问。对象所有者必须使用存储桶策略或 ACL 使该对象公开可读。
404 未找到	由于以下原因，网站端点的响应包含 404 Not Found： • Amazon S3 确定网站 URL 引用了不存在的对象键。 • Amazon S3 推断该请求针对不存在的索引文档。 • 在 URL 中指定的存储桶不存在。 • URL 中指定的存储桶存在，但未配置为网站。 您可以创建为 404 Not Found 返回的自定义文档。确保该文档已上传到配置为网站的存储桶，且网站托管配置已设置为使用该文档。 有关 Amazon S3 如何将 URL 当作对对象或索引文档的请求的信息，请参阅配置索引文档。

HTTP 错误代码	描述
500 Service Error	当出现内部服务器错误时，网站端点的响应包含 500 Service Error。
503 服务不可用	当 Amazon S3 确定您需要降低请求率时，网站端点的响应包含 503 Service Unavailable（503 服务不可用）。

对于其中每个错误，Amazon S3 都返回一条预定义的 HTML 消息。以下是对 403 Forbidden 响应返回的 HTML 消息示例。

403 Forbidden

- Code: AccessDenied
- Message: Access Denied
- RequestId: 873CA367A51F7EC7
- HostId: DdQezl9vkuw5luD5HKsFaTDm9KH4PZzCPRkW3igimILbTu1DiYlvXjgyd7pVxq32

An Error Occurred While Attempting to Retrieve a Custom Error Document

- Code: AccessDenied
- Message: Access Denied

配置自定义错误文档

将存储桶配置为静态网站时，您可以提供包含用户友好错误消息和其他帮助的自定义错误文档。Amazon S3 将仅为 HTTP 4XX 类的错误代码返回您的自定义错误文档。

要使用 S3 控制台配置自定义错误文档，请执行以下步骤。您还可以使用 REST API、AWS SDK、AWS CLI 或 AWS CloudFormation 配置错误文档。有关更多信息，请参阅下列内容：

- 《Amazon Simple Storage Service API 参考》中的 PutBucketWebsite
- 《AWS CloudFormation 用户指南》中的 AWS::S3::Bucket WebsiteConfiguration
- 《AWS CLI 命令参考》中的 put-bucket-website

当您为存储桶启用静态网站托管时，请输入错误文档的名称（例如，**404.html**）。为存储桶启用静态网站托管后，您可以将具有此错误文档名称的 HTML 文件上传到存储桶。

要配置错误文档

1. 创建错误文档，例如 404.html。
2. 将错误文档文件保存在本地。

   错误文档名称区分大小写，必须与启用静态网站托管时输入的名称完全匹配。例如，如果在静态网站托管对话框中为错误文档名称输入 404.html，则错误文档文件名也必须是 404.html。
3. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
4. 在存储桶 列表中，选择要用于托管静态网站的存储桶的名称。
5. 为您的存储桶启用静态网站托管，并输入错误文档的确切名称（例如 404.html）。有关更多信息，请参阅启用网站托管 和配置自定义错误文档。

   启用静态网站托管后，继续执行步骤 6。
6. 要将错误文档上传到存储桶，请执行以下操作之一：

   - 将错误文档文件拖放到控制台存储桶列表中。
   - 选择上传，然后按照提示选择并上传索引文件。

   如需分步指导，请参阅 上传对象。

设置访问网站的权限

将存储桶配置为静态网站时，如果希望网站成为公有网站，则可以授予公有读取访问权限。要使存储桶公开可读，您必须禁用存储桶的阻止公有访问设置并编写一条授予公有读取访问权限的存储桶策略。如果存储桶包含并非由存储桶拥有者拥有的对象，您可能还需要添加对象访问控制列表 (ACL)，以便向所有人授予读取访问权限。

如果您不想禁用存储桶的阻止公共访问设置，但仍希望将自己的网站公开，则可以创建 Amazon CloudFront 分配来为静态网站提供服务。有关更多信息，请参阅使用 Amazon CloudFront 为网站提速或《Amazon Route 53 开发人员指南》中的使用 Amazon CloudFront 分配为静态网站提供服务。

> **Note**
>
> 在网站端点上，如果用户请求了不存在的对象，则 Amazon S3 将返回 HTTP 响应代码 404 (Not Found)。如果该对象存在，但您尚未授予对对象的读取权限，则网站端点将返回 HTTP 响应代码 403 (Access Denied)。用户可以使用该响应代码推断特定对象是否存在。如果您不需要此行为，则不应启用对存储桶的网站支持。

主题

步骤 1：编辑 S3 阻止公有访问设置

如果要将现有存储桶配置为具有公有访问权限的静态网站，您必须编辑该存储桶的阻止公有访问设置。您可能还必须编辑账户级别的阻止公有访问设置。Amazon S3 会应用限制性最强的存储桶级别和账户级别的阻止公有访问设置的组合。

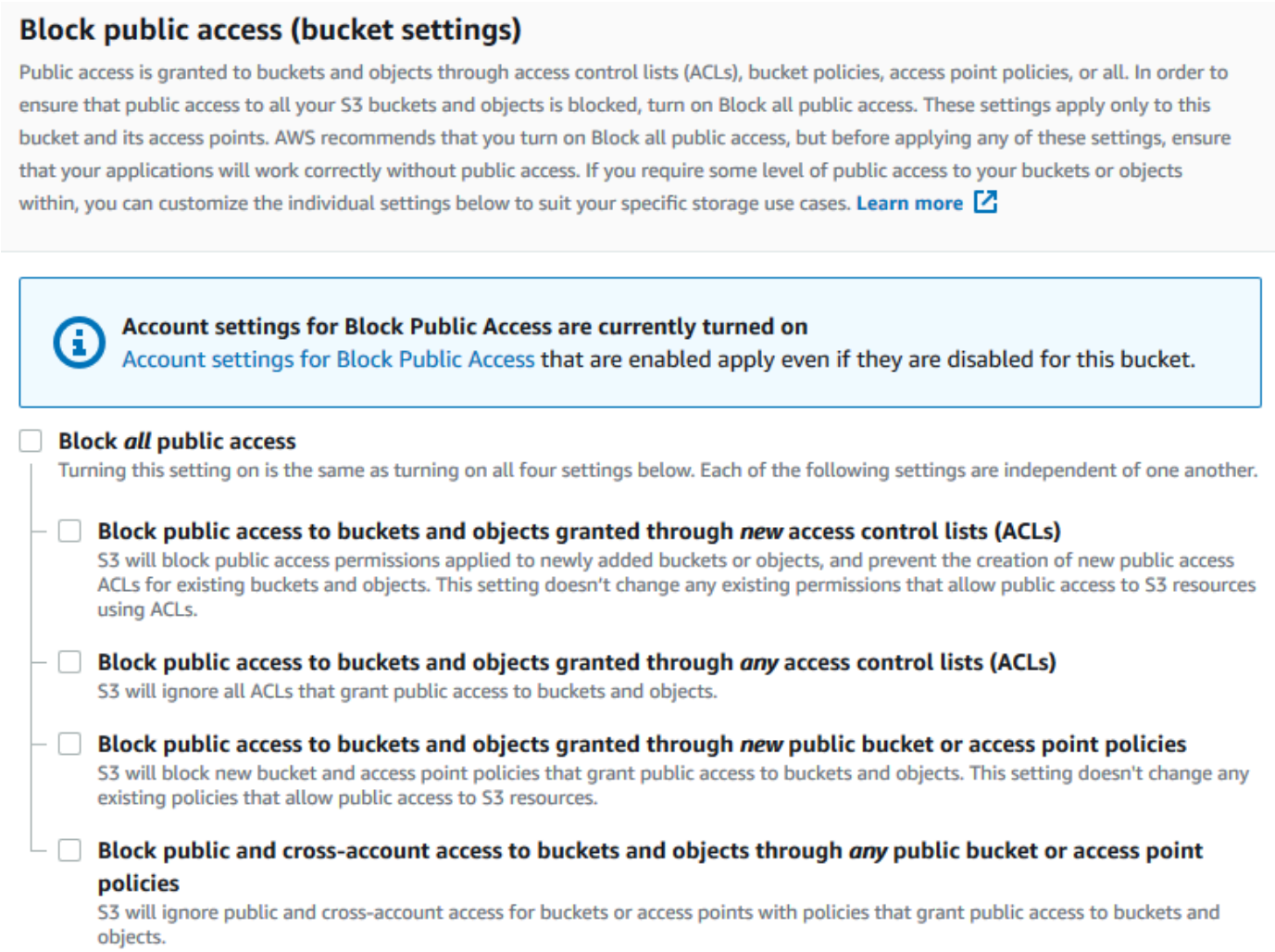
例如，如果您允许对存储桶进行公有访问，但阻止账户级别的所有公有访问权限，则 Amazon S3 将继续阻止对存储桶进行公有访问。在这种情况下，您将必须编辑存储桶级别和账户级别的阻止公有访问设置。有关更多信息，请参阅 阻止对您的 Amazon S3 存储的公有访问。

默认情况下，Amazon S3 阻止对您的账户和存储桶的公有访问权限。如果要使用存储桶托管静态网站，您可以使用以下步骤编辑您的屏蔽公共访问权限设置。

> **Warning**
>
> 在完成这些步骤之前，请查看阻止对您的 Amazon S3 存储的公有访问，来确保您了解并接受支持公共访问权限所涉及的风险。当您关闭屏蔽公共访问权限设置以使您的存储桶变为公有时，Internet 上的任何人都可以访问您的存储桶。我们建议您阻止对存储桶的所有公有访问。

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 请选择已配置为静态网站的存储桶的名称。
3. 选择权限。

4. 在屏蔽公共访问权限（存储桶设置）下，请选择编辑。
5. 清除阻止所有公有访问，然后选择保存更改。



Amazon S3 关闭了存储桶的屏蔽公共访问权限设置。要创建公有静态网站，可能还必须为您的账户配置屏蔽公共访问权限设置，然后再添加存储桶策略。如果当前已开启账户的屏蔽公共访问权限设置，您将在屏蔽公共访问权限（存储桶设置）下看到一条备注。

步骤 2：添加存储桶策略

要使存储桶中的对象公开可读，您必须编写一条向所有人授予 `s3:GetObject` 权限的存储桶策略。

在编辑 S3 阻止公有访问设置后，您可以添加存储桶策略以授予对存储桶的公有读取访问权限。当您授予公有读取访问权限时，Internet 上的任何人都可以访问您的存储桶。

> **Important**
>
> 下面的策略仅供举例说明，仍允许完全访问您存储桶的内容。在继续执行此步骤之前，请查看如何保护 Amazon S3 存储桶中的文件？，以确保您了解保护 S3 存储桶中文件的最佳实践以及授予公有访问权限所涉及的风险。

1. 在存储桶 下，请选择存储桶的名称。
2. 选择权限。
3. 在存储桶策略下，请选择编辑。
4. 要授予对网站的公有读取访问权限，请复制以下存储桶策略，将其粘贴到存储桶策略编辑器中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. 将 `Resource` 更新为您的存储桶名称。

   在上述示例存储桶策略中，*Bucket-Name* 是存储桶名称的占位符。要将此存储桶策略用于您自己的存储桶，您必须更新此名称以匹配您的存储桶名称。

6. 选择保存更改。

   此时将显示一条消息，指示存储桶策略已成功添加。

   如果您看到显示 `Policy has invalid resource` 的错误，请确认存储桶策略中的存储桶名称与您的存储桶名称匹配。有关添加存储桶策略的信息，请参阅如何添加 S3 存储桶策略？

如果您收到错误消息且无法保存存储桶策略，请检查您的账户和存储桶的阻止公有访问设置以确认您允许对存储桶进行公有访问。

对象访问控制列表

您可以使用存储桶策略来授予对对象的公共读取权限。但是，存储桶策略仅适用于存储桶拥有者所拥有的对象。如果存储桶包含并非由存储桶拥有者拥有的对象，则存储桶拥有者应使用对象访问控制列表 (ACL) 授予对这些对象的公有 READ 权限。

S3 对象所有权是 Amazon S3 存储桶级别的设置，您可以使用该设置来控制上传到存储桶的对象的所有权和禁用或启用 ACL。默认情况下，对象所有权设为强制存储桶拥有者设置，并且所有 ACL 均处于禁用状态。禁用 ACL 后，存储桶拥有者拥有存储桶中的所有对象，并使用访问管理策略来专门管理对这些对象的访问权限。

Amazon S3 中的大多数现代使用案例不再需要使用 ACL。我们建议您将 ACL 保持为禁用状态，除非有需要单独控制每个对象的访问权限的特殊情况。禁用 ACL 后，您可以使用策略来控制对存储桶中所有对象的访问权限，无论是谁将对象上传到您的存储桶。有关更多信息，请参阅 为您的存储桶控制对象所有权和禁用 ACL。。

> **Important**
>
> 如果您的存储桶针对 S3 对象所有权使用强制存储桶拥有者设置，则必须使用策略授予对存储桶及其中对象的访问权限。启用强制存储桶拥有者设置后，设置访问控制列表（ACL）或更新 ACL 的请求将失败并返回 `AccessControlListNotSupported` 错误代码。我们仍然支持读取 ACL 的请求。

要使用 ACL 使对象公开可读，您可以向 `AllUsers` 组授予 READ 权限，如以下授权元素所示。可以将此授权元素添加到对象 ACL 中。有关管理 ACL 的信息，请参阅 访问控制列表 (ACL) 概述。

```
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
```

（可选）记录 Web 流量

您可以选择为配置为静态网站的存储桶启用 Amazon S3 服务器访问日志记录。服务器访问日志记录详细地记录了对您的存储桶提出的各种请求。有关更多信息，请参阅 使用服务器访问日志记录来记录请求。如果您计划使用 Amazon CloudFront 来为网站提速，也可以使用 CloudFront 日志记录。有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的配置和使用访问日志。

为静态网站存储桶启用服务器访问日志记录

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 例如，在您创建存储桶（配置为静态网站）的同一区域中，创建用于日志记录的存储桶（例如 `logs.example.com`）。
3. 为服务器访问日志记录日志文件创建文件夹（例如，`logs`）。
4. （可选）如果要使用 CloudFront 提高网站性能，请为 CloudFront 日志文件创建一个文件夹（例如，`cdn`）。

   有关更多信息，请参阅 使用 Amazon CloudFront 为网站提速。
5. 在 Buckets (存储桶) 列表中，请选择您的存储桶。
6. 请选择属性。
7. 在 Server access logging（服务器访问日志记录）下，请选择 Edit（编辑）。
8. 请选择 Enable。
9. 在 Target bucket（目标存储桶）下，请选择服务器访问日志的存储桶和文件夹目标：
   - 浏览到文件夹和存储桶位置：
     1. 请选择 Browse S3（浏览 S3）。
     2. 请选择存储桶名称，然后选择日志文件夹。
     3. 请选择 Choose path（选择路径）。
   - 输入 S3 存储桶路径，例如 **`s3://logs.example.com/logs/`**。
10. 选择保存更改。

    在日志存储桶中，您现在可以访问日志。Amazon S3 每 2 小时将网站访问日志写入您的日志存储桶一次。

（可选）配置网页重定向

如果为静态网站托管配置了 Amazon S3 存储桶，您可以为存储桶或其中的对象配置重新导向。您可以使用以下选项来配置重新导向。

主题

针对存储桶的网站端点的请求重新导向到另一个存储桶或域

您可以将针对存储桶的网站端点的所有请求重新导向到另一个存储桶或域。如果您重新导向所有请求，则对网站端点所做的任何请求都将重新导向至指定的存储桶或域。

例如，假设您的根域为 `example.com`，而您需要服务 `http://example.com` 和 `http://www.example.com` 的请求，则必须创建两个分别名为 `example.com` 和 `www.example.com` 的存储桶。然后，将内容保留在 `example.com` 存储桶中，然后配置另一个 `www.example.com` 存储桶以将所有请求重定向至 `example.com` 存储桶。有关更多信息，请参阅使用自定义域名配置静态网站。

重定向对存储桶网站端点的请求

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）下，请选择要从中重定向请求的存储桶的名称（例如 `www.example.com`）。
3. 请选择属性。
4. 在静态网站托管下，选择编辑。
5. 请选择 Redirect requests for an object（重定向对于对象的请求）。
6. 在 Host name（主机名）框中，输入存储桶或自定义域的网站端点。

   例如，如果您正在重定向到根域地址，则输入 **`example.com`**。
7. 对于 Protocol（协议），请选择重定向请求的协议（none（无）、http 或 https）。

   如果未指定协议，则默认选项为 none (无)。
8. 选择保存更改。

配置重新导向规则以使用高级条件重新导向

利用高级重定向规则，您可以根据特定对象键名、请求中的前缀或者响应代码来按条件路由请求。例如，假设您在存储桶中删除或重命名了某个对象。您可以添加一个将请求重定向到其他对象的路由规则。如果您要使文件夹不可用，则可以添加路由规则将请求重定向至其他网页。处理错误时，您还可以通过将返回错误的请求路由到其他域来添加一个处理错误条件的路由规则。

当为存储桶启用静态网站托管时，您可以选择指定高级重新导向规则。Amazon S3 对每个网站配置的路由规则限制为 50 条。如果您需要 50 个以上的路由规则，则可以使用对象重定向。有关更多信息，请参阅 使用 S3 控制台。

有关使用 REST API 配置路由规则的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketWebsite。

> **Important**
>
> 要在新的 Amazon S3 控制台中创建重定向规则，您必须使用 JSON。有关 JSON 示例，请参阅重定向规则示例。

为静态网站配置重定向规则

要为已启用静态网站托管的存储桶添加重定向规则，请执行以下步骤。

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Bucket（存储桶）列表中，请选择已配置为静态网站的存储桶的名称。
3. 请选择属性。
4. 在 Static website hosting（静态网站托管）下，请选择 Edit (编辑)。
5. 在 Redirection rules (重定向规则) 框中，输入使用 JSON 的重定向规则。

   在 S3 控制台中，您可以使用 JSON 描述规则。有关 JSON 示例，请参阅重定向规则示例。Amazon S3 对每个网站配置的路由规则限制为 50 条。
6. 选择保存更改。

路由规则元素

以下是在 JSON 和 XML 中的网站配置中定义路由规则的一般语法。要在新的 S3 控制台中配置重定向规则，必须使用 JSON。有关 JSON 示例，请参阅重定向规则示例。

JSON

```
[
    {
      "Condition": {
        "HttpErrorCodeReturnedEquals": "string",
        "KeyPrefixEquals": "string"
      },
      "Redirect": {
        "HostName": "string",
        "HttpRedirectCode": "string",
        "Protocol": "http"|"https",
        "ReplaceKeyPrefixWith": "string",
        "ReplaceKeyWith": "string"
      }
    }
  ]

Note: Redirect must each have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

XML

```
<RoutingRules> =
    <RoutingRules>
         <RoutingRule>...</RoutingRule>
         [<RoutingRule>...</RoutingRule>
         ...]
    </RoutingRules>

<RoutingRule> =
   <RoutingRule>
      [ <Condition>...</Condition> ]
      <Redirect>...</Redirect>
   </RoutingRule>

<Condition> =
   <Condition>
      [ <KeyPrefixEquals>...</KeyPrefixEquals> ]
      [ <HttpErrorCodeReturnedEquals>...</HttpErrorCodeReturnedEquals> ]
   </Condition>
    Note: <Condition> must have at least one child element.
```

```
<Redirect> =
   <Redirect>
      [ <HostName>...</HostName> ]
      [ <Protocol>...</Protocol> ]
      [ <ReplaceKeyPrefixWith>...</ReplaceKeyPrefixWith>  ]
      [ <ReplaceKeyWith>...</ReplaceKeyWith> ]
      [ <HttpRedirectCode>...</HttpRedirectCode> ]
   </Redirect>

Note: <Redirect> must have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

下表描述了路由规则中的元素。

名称	描述
RoutingRules	用于收集 RoutingRule 元素的容器。
RoutingRule	标识在满足条件时应用的条件和重定向的规则。 条件： • RoutingRules 容器必须包含至少一个路由规则。
Condition	对于要应用的指定的重定向，用于描述必须满足的条件的容器。如果路由规则不包含条件，则该规则将应用于所有请求。
KeyPrefixEquals	从中重定向请求的对象键名的前缀。 KeyPrefixEquals 如果未指定 HttpErrorCodeReturnedEquals ，则需要 。如果同时指定 KeyPrefixEquals 和 HttpErrorCodeReturnedEquals ，则两者都必须为真才能满足条件。
HttpErrorCodeReturnedEquals	

名称	描述
	要应用的重定向必须匹配的 HTTP 错误代码。如果出现错误，并且错误代码满足此值，则应用指定的重定向。 HttpErrorCodeReturnedEquals 如果未指定 KeyPrefixEquals ，则需要 。如果同时指定 KeyPrefixEquals 和 HttpErrorCodeReturnedEquals ，则两者都必须为真才能满足条件。
Redirect	提供用于重定向请求的说明的容器元素。您可以将请求重定向到其他主机或其他页面，或者也可以指定要使用的其他协议。RoutingRule 必须具有 Redirect 元素。Redirect 元素必须至少包含以下一个同级元素：Protocol、HostName、ReplaceKeyPrefixWith 、ReplaceKeyWith 或 HttpRedirectCode 。
Protocol	响应中返回的 http 标头中将要使用的协议（https 或 Location）。 如果提供了一个同级，则 Protocol 不是必需的。
HostName	可在响应中返回的 Location 标头中使用的主机名。 如果提供了一个同级，则 HostName 不是必需的。
ReplaceKeyPrefixWith	将替换重定向请求中的 KeyPrefixEquals 值的对象键名的前缀。 如果提供了一个同级，则 ReplaceKeyPrefixWith 不是必需的。仅在不提供 ReplaceKeyWith 时提供它。
ReplaceKeyWith	可在响应中返回的 Location 标头中使用的对象键。 如果提供了一个同级，则 ReplaceKeyWith 不是必需的。仅在不提供 ReplaceKeyPrefixWith 时提供它。

名称	描述
HttpRedirectCode	可在响应中返回的 Location 标头中使用的 HTTP 重定向代码。 如果提供了一个同级，则 HttpRedirectCode 不是必需的。

重定向规则示例

以下示例介绍了常见的重定向任务：

> **Important**
>
> 要在新的 Amazon S3 控制台中创建重定向规则，您必须使用 JSON。

Example 1：保留键前缀后进行重定向

假设您的存储桶包含以下对象：

- index.html
- docs/article1.html
- docs/article2.html

您决定将文件夹从 docs/ 重命名为 documents/。做出此更改后，您需要将对前缀 docs/ 的请求重定向至 documents/。例如，对 docs/article1.html 的请求将重定向至 documents/article1.html。

在这种情况下，您可以将以下路由规则添加到网站配置。

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "docs/"
        },
        "Redirect": {
            "ReplaceKeyPrefixWith": "documents/"
```

```
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <KeyPrefixEquals>docs/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

Example 2：将对已删除文件夹的请求重定向到页面

假设您删除了 images/ 文件夹 (即，删除了具有键前缀 images/ 的所有对象)。您可以添加路由规则以将具有键前缀 images/ 的任何对象的请求重定向至名为 folderdeleted.html 的页。

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "images/"
        },
        "Redirect": {
            "ReplaceKeyWith": "folderdeleted.html"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
       <KeyPrefixEquals>images/</KeyPrefixEquals>
```

```
    </Condition>
    <Redirect>
      <ReplaceKeyWith>folderdeleted.html</ReplaceKeyWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

Example 3：为 HTTP 错误进行重定向

假设在未找到请求的对象时，您需要将请求重定向到 Amazon Elastic Compute Cloud (Amazon EC2) 实例。添加重定向规则，以便当返回 HTTP 状态代码 404 (未找到) 时，站点访问者可重定向到将处理该请求的 Amazon EC2 实例。

以下示例也将在重定向中插入对象键前缀 `report-404/`。例如，如果您请求了页面 `ExamplePage.html` 并且它导致了 HTTP 404 错误，该请求将重定向到指定 Amazon EC2 实例上的 `report-404/ExamplePage.html` 页面。如果没有路由规则且发生了 HTTP 错误 404，将返回配置中指定的错误文档。

JSON

```
[
    {
        "Condition": {
            "HttpErrorCodeReturnedEquals": "404"
        },
        "Redirect": {
            "HostName": "ec2-11-22-333-44.compute-1.amazonaws.com",
            "ReplaceKeyPrefixWith": "report-404/"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals >
    </Condition>
    <Redirect>
      <HostName>ec2-11-22-333-44.compute-1.amazonaws.com</HostName>
```

```
      <ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

重新导向对于对象的请求

您可以通过在对象的元数据中设置网站重定向位置，以将对于对象的请求重定向到另一个对象或 URL。您可以通过将 `x-amz-website-redirect-location` 属性添加到对象元数据来设置重定向。在 Amazon S3 控制台上，您可以在对象的元数据中设置 Website Redirect Location (网站重定向位置)。如果您使用 Amazon S3 API，请设置 `x-amz-website-redirect-location`。然后，该网站将该对象解释为 301 重定向。

要重定向对其他对象的请求，您可以将重定向位置设置为目标对象的键。要重定向对外部 URL 的请求，您可以将重定向位置设置为所需的 URL。有关对象元数据的更多信息，请参阅 系统定义的对象元数据。

当您设置页面重定向时，您可以保留或删除源对象内容。例如，如果您的存储桶中有一个 `page1.html` 对象，则可以将针对此页面的任何请求重定向到另一个对象 `page2.html`。您有两种选择：

- 保留 `page1.html` 对象的内容和重定向页面请求。
- 删除 `page1.html` 的内容并上传名为 `page1.html` 的零字节对象以替换现有对象和重定向页面请求。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择已配置为静态网站的存储桶的名称（例如 `example.com`）。
3. 在 Objects (对象) 下，选择您的对象。
4. 请选择 Actions（操作），然后选择 Edit metadata（编辑元数据）。
5. 请选择 Metadata（元数据）。
6. 请选择 Add Metadata（添加元数据）。
7. 在 Type（类型）下，请选择 System Defined（系统定义）。
8. 在 Key (键) 中，请选择 x-amz-website-redirect-location。

9. 在 Value (值)中，输入要重定向到的对象的键名，例如 `/page2.html`。

   对于同一存储桶中的另一个对象，值中的 `/` 前缀是必需的。您也可以将该值设置为外部 URL，例如 `http://www.example.com`。

10. 请选择 Edit metadata（编辑元数据）。

使用 REST API

以下 Amazon S3 API 操作支持请求中的 `x-amz-website-redirect-location` 标头。Amazon S3 存储对象元数据中的标头值作为 `x-amz-website-redirect-location`。

- PUT Object
- 开始分段上传
- POST 对象
- PUT Object – 复制

为网站托管配置的存储桶具有网站端点和 REST 端点。对配置为 301 重定向的页面的请求具有以下可能的结果，具体取决于请求的端点：

- 特定于区域的网站端点 – Amazon S3 根据 `x-amz-website-redirect-location` 属性的值重定向页面请求。
- REST 端点 – Amazon S3 不会重定向页面请求。它将返回请求的对象。

有关端点的更多信息，请参阅 网站端点和 REST API 端点之间的主要区别。

设置页面重定向时，您可以保留或删除对象内容。例如，假设您的存储桶中有 `page1.html` 对象。

- 若要保留 `page1.html` 的内容并仅重定向页面请求，您可以提交 PUT 对象 – 复制请求来创建新的 `page1.html` 对象，该对象使用现有 `page1.html` 对象作为源。在您的请求中，您可以设置 `x-amz-website-redirect-location` 标头。当请求完成时，您将拥有未更改其内容的原始页面，但 Amazon S3 会将对该页面的任何请求重定向到您指定的重定向位置。
- 要删除 `page1.html` 对象的内容并重定向对该页面的请求，您可以发送 PUT Object 请求，以上传具有相同对象键的零字节对象 `page1.html`。在 PUT 请求中，您可以将 `x-amz-website-redirect-location` 的 `page1.html` 设置为新对象。当请求完成时，`page1.html` 将不包含任何内容，且后续请求将被重定向到由 `x-amz-website-redirect-location` 指定的位置。

当使用 GET Object 操作以及其他对象元数据检索该对象时，Amazon S3 在响应中返回 `x-amz-website-redirect-location` 标头。

使用跨源资源共享 (CORS)

跨源资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。借助 CORS 支持，您可以使用 Amazon S3 构建各种富客户端 Web 应用程序，并选择性地允许跨源访问您的 Amazon S3 资源。

本部分提供 CORS 概述。副主题介绍如何通过使用 Amazon S3 控制台或通过以编程方式使用 Amazon S3 REST API 和 AWS SDK 来启用 CORS。

跨源资源共享：使用案例场景

以下是有关使用 CORS 的示例场景。

方案 1

假设您在名为 `website` 的 Amazon S3 存储桶中托管网站（如 使用 Amazon S3 托管静态网站 中所述）。您的用户加载了网站端点。

```
http://website.s3-website.us-east-1.amazonaws.com
```

现在，您想要使用此存储桶中存储的网页上的 JavaScript，以使用该存储桶的 Amazon S3 API 端点 `website.s3.us-east-1.amazonaws.com` 向同一存储桶发送经身份验证的 GET 和 PUT 请求。浏览器通常会阻止 JavaScript 允许这些请求，但借助 CORS，您可以配置您的存储桶以显式支持来自 `website.s3-website.us-east-1.amazonaws.com` 的跨源请求。

方案 2

假设您想要托管来自您的 S3 存储桶的 Web 字体。浏览器会再次要求对正在加载的 Web 字体进行 CORS 检查 (也称为预检)。您可以配置托管 Web 字体的存储桶，以允许任何源发出这些请求。

Amazon S3 如何评估针对存储桶的 CORS 配置？

Amazon S3 收到来自浏览器的预检请求后，它将为存储桶评估 CORS 配置，并使用第一个匹配传入浏览器请求的 `CORSRule` 规则来允许跨源请求。要使规则实现匹配，必须满足以下条件：

- 向存储桶发出的 CORS 请求中的 `Origin` 标头必须与 CORS 配置的 `AllowedOrigins` 元素中的源相匹配。

- 在向存储桶发出的 CORS 请求的 `Access-Control-Request-Method` 中指定的 HTTP 方法，必须与在 CORS 配置的 `AllowedMethods` 元素中列出的一个或多个方法相匹配。
- 预检请求的 `Access-Control-Request-Headers` 标头中列出的标头必须与 CORS 配置的 `AllowedHeaders` 元素中的标头相匹配。

> **Note**
>
> 对存储桶启用 CORS 时，ACL 和策略将继续适用。

对象 Lambda 接入点如何支持 CORS

当 S3 对象 Lambda 收到来自浏览器的请求或包含 `Origin` 标头的请求时，S3 对象 Lambda 始终会添加 `“AllowedOrigins":"*"` 标头字段。

有关使用 CORS 的更多信息，请参阅以下主题。

主题

- CORS 配置的元素
- 配置跨源资源共享 (CORS)
- CORS 问题排查

CORS 配置的元素

要将您的存储桶配置为允许跨源请求，创建一个 CORS 配置。CORS 配置是一个包含元素的文档，这些元素标识可访问您的存储桶的源、对于每个源将支持的操作（HTTP 方法）以及其它操作特定的信息。您可以向配置添加最多 100 条规则。您可以将 CORS 配置作为 `cors` 子资源添加到存储桶中

如果要在 S3 控制台中配置 CORS，则必须使用 JSON 来创建 CORS 配置。新的 S3 控制台仅支持 JSON CORS 配置。

有关 CORS 配置及其中元素的更多信息，请参阅以下主题。有关如何添加 CORS 配置的说明，请参阅配置跨源资源共享 (CORS)。

> **Important**
>
> 在 S3 控制台中，CORS 配置必须是 JSON。

主题

AllowedMethods 元素

在 CORS 配置中，您可以为 AllowedMethods 元素指定以下值。

- GET
- PUT
- POST
- DELETE
- HEAD

AllowedOrigins 元素

在 AllowedOrigins 元素中，可指定您希望允许从中发送跨源请求的源，例如 http://www.example.com。源字符串只能包含至少一个 * 通配符，例如 http://*.example.com。您可以选择将 * 指定为源，以允许所有源发送跨源请求。您还可以指定 https 只允许安全的源。

AllowedHeaders 元素

AllowedHeaders 元素通过 Access-Control-Request-Headers 标头指定预检请求中允许哪些标头。Access-Control-Request-Headers 标头中的每个标头名称必须匹配元素中的相应条目。Amazon S3 将仅发送请求的响应中允许的标头。有关适用于发送至 Amazon S3 的请求中的标头示例列表，请参阅 Amazon Simple Storage Service API 参考指南中的常见请求标头。

配置中的每个 AllowedHeaders 字符串都可以包含至少一个 * 通配符。例如，<AllowedHeader>x-amz-*</AllowedHeader> 将允许所有特定于 Amazon 的标头。

`ExposeHeaders` 元素

每个 `ExposeHeader` 元素标识您希望客户能够从其应用程序 (例如，从 JavaScript `XMLHttpRequest` 对象) 进行访问的响应标头。有关常见的 Amazon S3 响应标头的列表，请参阅 Amazon Simple Storage Service API 参考指南中的常见响应标头。

`MaxAgeSeconds` 元素

`MaxAgeSeconds` 元素指定在预检请求被资源、HTTP 方法和源识别之后，浏览器将为预检请求缓存响应的时间 (以秒为单位)。

CORS 配置的示例

您可以使用您自己的域 (例如 `example1.com`) 提供您的内容，而不是通过使用 Amazon S3 网站端点访问网站。有关如何使用您自己的域的信息，请参阅 教程：使用注册到 Route 53 的自定义域配置静态网站。

以下示例 CORS 配置具有三个规则，这些规则被指定为 `CORSRule` 元素：

- 第一个规则允许来自 `http://www.example1.com` 源的跨源 PUT、POST 和 DELETE 请求。该规则还通过 `Access-Control-Request-Headers` 标头允许预检 OPTIONS 请求中的所有标头。作为对预检 OPTIONS 请求的响应，Amazon S3 将返回请求的标头。
- 第二个规则允许与第一个规则具有相同的跨源请求，但第二个规则应用于另一个源 `http://www.example2.com`。
- 第三个规则允许来自所有源的跨源 GET 请求。* 通配符将引用所有源。

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example1.com"
```

```
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example2.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example1.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>http://www.example2.com</AllowedOrigin>
```

```
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedMethod>DELETE</AllowedMethod>

    <AllowedHeader>*</AllowedHeader>
  </CORSRule>
  <CORSRule>
    <AllowedOrigin>*</AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
  </CORSRule>
</CORSConfiguration>
```

CORS 配置还允许可选的配置参数，如下面的 CORS 配置所示。在本示例中，CORS 配置允许来自 `http://www.example.com` 源的跨源 PUT、POST 和 DELETE 请求。

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example.com"
        ],
        "ExposeHeaders": [
            "x-amz-server-side-encryption",
            "x-amz-request-id",
            "x-amz-id-2"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example.com</AllowedOrigin>
   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
  <MaxAgeSeconds>3000</MaxAgeSeconds>
  <ExposeHeader>x-amz-server-side-encryption</
ExposeHeader>
  <ExposeHeader>x-amz-request-id</
ExposeHeader>
  <ExposeHeader>x-amz-id-2</ExposeHeader>
 </CORSRule>
</CORSConfiguration>
```

上述配置中的 CORSRule 元素包括以下可选元素：

- MaxAgeSeconds — 指定在 Amazon S3 针对特定资源的预检 OPTIONS 请求作出响应后，浏览器缓存该响应的时间（以秒为单位，在本示例中为 3000 秒）。通过缓存响应，在需要重复原始请求时，浏览器无需向 Amazon S3 发送预检请求。
- ExposeHeader – 识别可让客户从应用程序 (例如，从 JavaScript x-amz-server-side-encryption 对象) 进行访问的响应标头 (在本示例中，为 x-amz-request-id、x-amz-id-2 和 XMLHttpRequest)。

配置跨源资源共享 (CORS)

跨源资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。借助 CORS 支持，您可以使用 Amazon S3 构建各种富客户端 Web 应用程序，并选择性地允许跨源访问您的 Amazon S3 资源。

本节将向您介绍如何使用 Amazon S3 控制台、Amazon S3 REST API 和 AWS SDK 来启用 CORS。要将您的存储桶配置为允许跨源请求，您可以将 CORS 配置添加到存储桶中。CORS 配置是一个定义规则的文档，这些规则标识可访问您的存储桶的源、每个源支持的操作（HTTP 方法）以及其他操作特定的信息。在 S3 控制台中，CORS 配置必须是 JSON 文档。

有关 JSON 和 XML 中的 CORS 配置示例，请参阅 CORS 配置的元素。

使用 S3 控制台

本节说明如何使用 Amazon S3 控制台向 S3 存储桶添加跨源资源共享（CORS）配置。

在存储桶上启用 CORS 时，访问控制列表（ACL）和其他访问权限策略仍适用。

> **Important**
>
> 在 S3 控制台中，CORS 配置必须是 JSON。有关 JSON 和 XML 中的 CORS 配置示例，请参阅 CORS 配置的元素。

将 CORS 配置添加到 S3 存储桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择要为其创建存储桶策略的存储桶的名称。
3. 选择 Permissions (权限)。
4. 在 Cross-origin resource sharing（CORS）（跨源资源共享（CORS））部分中，请选择 Edit（编辑）。
5. 在 CORS configuration editor (CORS 配置编辑器) 文本框中，键入或复制并粘贴新的 CORS 配置，或者编辑现有配置。

   CORS 配置是一个 JSON 文件。您在编辑器中键入的文本必须是有效的 JSON。有关更多信息，请参阅 CORS 配置的元素。
6. 请选择保存更改。

> **Note**
>
> Amazon S3 在 CORS configuration editor（CORS 配置编辑器）标题旁边显示存储桶的 Amazon Resource Name（ARN）。有关 ARN 的更多信息，请参阅《Amazon Web Services 一般参考》中的 Amazon 资源名称（ARN）和 AWS 服务命名空间。

使用 AWS SDK

您可以使用 AWS SDK 管理存储桶的跨源资源共享（CORS）。有关 CORS 的更多信息，请参阅 使用跨源资源共享 (CORS)。

以下示例：

- 创建 CORS 配置并对存储桶设置该配置
- 通过添加规则来检索并修改配置
- 向存储桶添加修改过的配置
- 删除配置

Java

Example

Example

有关如何创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketCrossOriginConfiguration;
import com.amazonaws.services.s3.model.CORSRule;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CORS {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        // Create two CORS rules.
        List<CORSRule.AllowedMethods> rule1AM = new
 ArrayList<CORSRule.AllowedMethods>();
        rule1AM.add(CORSRule.AllowedMethods.PUT);
        rule1AM.add(CORSRule.AllowedMethods.POST);
        rule1AM.add(CORSRule.AllowedMethods.DELETE);
```

```
        CORSRule rule1 = new
CORSRule().withId("CORSRule1").withAllowedMethods(rule1AM)
                .withAllowedOrigins(Arrays.asList("http://*.example.com"));

        List<CORSRule.AllowedMethods> rule2AM = new
ArrayList<CORSRule.AllowedMethods>();
        rule2AM.add(CORSRule.AllowedMethods.GET);
        CORSRule rule2 = new
CORSRule().withId("CORSRule2").withAllowedMethods(rule2AM)
                .withAllowedOrigins(Arrays.asList("*")).withMaxAgeSeconds(3000)
                .withExposedHeaders(Arrays.asList("x-amz-server-side-encryption"));

        List<CORSRule> rules = new ArrayList<CORSRule>();
        rules.add(rule1);
        rules.add(rule2);

        // Add the rules to a new CORS configuration.
        BucketCrossOriginConfiguration configuration = new
BucketCrossOriginConfiguration();
        configuration.setRules(rules);

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Add the configuration to the bucket.
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Retrieve and display the configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);

            // Add another new rule.
            List<CORSRule.AllowedMethods> rule3AM = new
ArrayList<CORSRule.AllowedMethods>();
            rule3AM.add(CORSRule.AllowedMethods.HEAD);
            CORSRule rule3 = new
CORSRule().withId("CORSRule3").withAllowedMethods(rule3AM)
                    .withAllowedOrigins(Arrays.asList("http://www.example.com"));

            rules = configuration.getRules();
            rules.add(rule3);
```

```
            configuration.setRules(rules);
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Verify that the new rule was added by checking the number of rules in
the
            // configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            System.out.println("Expected # of rules = 3, found " +
configuration.getRules().size());

            // Delete the configuration.
            s3Client.deleteBucketCrossOriginConfiguration(bucketName);
            System.out.println("Removed CORS configuration.");

            // Retrieve and display the configuration to verify that it was
            // successfully deleted.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void printCORSConfiguration(BucketCrossOriginConfiguration
configuration) {
        if (configuration == null) {
            System.out.println("Configuration is null.");
        } else {
            System.out.println("Configuration has " +
configuration.getRules().size() + " rules\n");

            for (CORSRule rule : configuration.getRules()) {
                System.out.println("Rule ID: " + rule.getId());
                System.out.println("MaxAgeSeconds: " + rule.getMaxAgeSeconds());
                System.out.println("AllowedMethod: " + rule.getAllowedMethods());
                System.out.println("AllowedOrigins: " + rule.getAllowedOrigins());
                System.out.println("AllowedHeaders: " + rule.getAllowedHeaders());
                System.out.println("ExposeHeader: " + rule.getExposedHeaders());
```

```
                System.out.println();
            }
        }
    }
}
```

.NET

Example

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CORSTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CORSConfigTestAsync().Wait();
        }
        private static async Task CORSConfigTestAsync()
        {
            try
            {
                // Create a new configuration request and add two rules
                CORSConfiguration configuration = new CORSConfiguration
                {
                    Rules = new System.Collections.Generic.List<CORSRule>
```

```
                        {
                          new CORSRule
                          {
                            Id = "CORSRule1",
                            AllowedMethods = new List<string> {"PUT", "POST",
 "DELETE"},
                            AllowedOrigins = new List<string> {"http://
*.example.com"}
                          },
                          new CORSRule
                          {
                            Id = "CORSRule2",
                            AllowedMethods = new List<string> {"GET"},
                            AllowedOrigins = new List<string> {"*"},
                            MaxAgeSeconds = 3000,
                            ExposeHeaders = new List<string> {"x-amz-server-side-
encryption"}
                          }
                        }
                };

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Retrieve an existing configuration.
                configuration = await RetrieveCORSConfigurationAsync();

                // Add a new rule.
                configuration.Rules.Add(new CORSRule
                {
                    Id = "CORSRule3",
                    AllowedMethods = new List<string> { "HEAD" },
                    AllowedOrigins = new List<string> { "http://www.example.com" }
                });

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Verify that there are now three rules.
                configuration = await RetrieveCORSConfigurationAsync();
                Console.WriteLine();
                Console.WriteLine("Expected # of rulest=3; found:{0}",
 configuration.Rules.Count);
                Console.WriteLine();
```

```
                Console.WriteLine("Pause before configuration delete. To continue,
click Enter...");
                Console.ReadKey();

                // Delete the configuration.
                await DeleteCORSConfigurationAsync();

                // Retrieve a nonexistent configuration.
                configuration = await RetrieveCORSConfigurationAsync();
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task PutCORSConfigurationAsync(CORSConfiguration configuration)
        {

            PutCORSConfigurationRequest request = new PutCORSConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };

            var response = await s3Client.PutCORSConfigurationAsync(request);
        }

        static async Task<CORSConfiguration> RetrieveCORSConfigurationAsync()
        {
            GetCORSConfigurationRequest request = new GetCORSConfigurationRequest
            {
                BucketName = bucketName

            };
            var response = await s3Client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
```

```
            return configuration;
        }

        static async Task DeleteCORSConfigurationAsync()
        {
            DeleteCORSConfigurationRequest request = new
 DeleteCORSConfigurationRequest
            {
                BucketName = bucketName
            };
            await s3Client.DeleteCORSConfigurationAsync(request);
        }

        static void PrintCORSRules(CORSConfiguration configuration)
        {
            Console.WriteLine();

            if (configuration == null)
            {
                Console.WriteLine("\nConfiguration is null");
                return;
            }

            Console.WriteLine("Configuration has {0} rules:",
 configuration.Rules.Count);
            foreach (CORSRule rule in configuration.Rules)
            {
                Console.WriteLine("Rule ID: {0}", rule.Id);
                Console.WriteLine("MaxAgeSeconds: {0}", rule.MaxAgeSeconds);
                Console.WriteLine("AllowedMethod: {0}", string.Join(", ",
 rule.AllowedMethods.ToArray()));
                Console.WriteLine("AllowedOrigins: {0}", string.Join(", ",
 rule.AllowedOrigins.ToArray()));
                Console.WriteLine("AllowedHeaders: {0}", string.Join(", ",
 rule.AllowedHeaders.ToArray()));
                Console.WriteLine("ExposeHeader: {0}", string.Join(", ",
 rule.ExposeHeaders.ToArray()));
            }
        }
    }
}
```

使用 REST API

要对存储桶设置 CORS 配置，您可以使用 AWS Management Console。如果您的应用程序需要它，您也可以直接发送 REST 请求。Amazon Simple Storage Service API 参考的下面几节描述了与 CORS 配置相关的 REST API 操作：

- PutBucketCors
- GetBucketCors
- DeleteBucketCors
- OPTIONS object

CORS 问题排查

以下主题有助于排查与 S3 相关的一些常见 CORS 问题。

主题

403 禁止错误：未为此存储桶启用 CORS

当跨源请求发送到 Amazon S3 但未在 S3 存储桶上配置 CORS 时，会发生以下 `403 Forbidden` 错误。

错误：HTTP/1.1 403 禁止 CORS 响应：未为此存储桶启用 CORS。

CORS 配置是一个包含规则的文档或策略，这些规则标识可访问您的存储桶的源、每个源将支持的操作（HTTP 方法）以及其它操作特定的信息。请参阅如何使用 Amazon S3 控制台、AWS SDK 和 REST API 在 S3 上配置 CORS。有关 CORS 的更多信息以及 CORS 配置示例，请参阅 CORS 的元素。

403 禁止错误：不支持此 CORS 请求

当 CORS 配置中的 CORS 规则与请求中的数据不匹配时，会收到以下 `403 Forbidden` 错误。

错误：HTTP/1.1 403 禁止 CORS 响应：不支持此 CORS 请求。

因此，出现此 403 Forbidden 错误的原因可能有多种：

- 不支持源。
- 不支持方法。
- 不支持请求的标头。

对于 Amazon S3 收到的每个请求，CORS 配置中必须具有与请求中的数据匹配的 CORS 规则。

不支持源

向存储桶发出的 CORS 请求中的 Origin 标头必须与 CORS 配置的 AllowedOrigins 元素中的源相匹配。AllowedOrigins 元素中的通配符 ("*") 将匹配所有 HTTP 方法。有关如何更新 AllowedOrigins 元素的更多信息，请参阅配置跨源资源共享（CORS）。

例如，如果 AllowedOrigins 元素中仅包含 http://www.example1.com 域，则从 http://www.example2.com 域发送的 CORS 请求将收到 403 Forbidden 错误。

以下示例显示 CORS 配置的一部分，它在 AllowedOrigins 元素中包含 http://www.example1.com 域。

```
"AllowedOrigins":[
   "http://www.example1.com"
]
```

要使从 http://www.example2.com 域发送的 CORS 请求获得成功，应将 http://www.example2.com 域包含在 CORS 配置的 AllowedOrigins 元素中。

```
"AllowedOrigins":[
   "http://www.example1.com"
   "http://www.example2.com"
]
```

不支持方法

在向存储桶发出的 CORS 请求的 Access-Control-Request-Method 中指定的 HTTP 方法，必须与在 CORS 配置的 AllowedMethods 元素中列出的一个或多个方法相匹配。AllowedMethods 中的通配符 ("*") 将匹配所有 HTTP 方法。有关如何更新 AllowedOrigins 元素的更多信息，请参阅配置跨源资源共享（CORS）。

在 CORS 配置中，您可以在 AllowedMethods 元素中指定以下方法：

- GET
- PUT
- POST
- DELETE
- HEAD

以下示例显示 CORS 配置的一部分，该配置在 AllowedMethods 元素中包含 GET 方法。只有包含 GET 方法的请求才会获得成功。

```
"AllowedMethods":[
   "GET"
]
```

如果在 CORS 请求中使用了 HTTP 方法（例如 PUT），或者该方法包含在对存储桶的预检 CORS 请求中但不存在于 CORS 配置中，则该请求将导致 403 Forbidden 错误。要支持此 CORS 请求或 CORS 预检请求，必须将 PUT 方法添加到 CORS 配置中。

```
"AllowedMethods":[
   "GET"
   "PUT"
]
```

不支持请求的标头

预检请求的 Access-Control-Request-Headers 标头中列出的标头必须与 CORS 配置的 AllowedHeaders 元素中的标头相匹配。有关可以在对 Amazon S3 的请求中使用的常见标头的列表，请参阅 Common Request Headers。有关如何更新 AllowedHeaders 元素的更多信息，请参阅配置跨源资源共享（CORS）。

以下示例显示 CORS 配置的一部分，该配置在 AllowedHeaders 元素中包含 Authorization 标头。只有对 Authorization 标头的请求才会获得成功。

```
"AllowedHeaders":  [
    "Authorization"
]
```

如果 CORS 请求中包含标头（例如 Content-MD5），但该标头不存在于 CORS 配置中，则该请求将导致 403 Forbidden 错误。要支持此 CORS 请求，必须将 Content-MD5 标头添加到 CORS 配置

中。如果您要在 CORS 请求中同时将 Authorization 和 Content-MD5 标头传递到存储桶，请确认这两个标头都包含在 CORS 配置的 AllowedHeaders 元素中。

```
"AllowedHeaders":  [
    "Authorization"
    "Content-MD5"
]
```

在 CORS 响应中找不到标头

CORS 配置中的 ExposeHeaders 元素确定了您希望在浏览器中运行的脚本和应用程序可以访问哪些响应标头，以便响应 CORS 请求。

如果存储在 S3 存储桶中的对象具有用户定义的元数据（例如 x-amz-meta-custom-header）以及响应数据，则此自定义标头可能包含您要从客户端 JavaScript 代码访问的其它元数据或信息。但出于安全考虑，默认情况下，浏览器会阻止访问自定义标头。要支持客户端 JavaScript 访问自定义标头，需要在 CORS 配置中包含该标头。

在下面的示例中，x-amz-meta-custom-header1 标头包含在 ExposeHeaders 元素中。x-amz-meta-custom-header2 未包含在 ExposeHeaders 元素中，并且 CORS 配置中也缺少此标头。在响应中，仅返回 ExposeHeaders 元素中包含的值。如果请求在 Access-Control-Expose-Headers 标头中包含了 x-amz-meta-custom-header2 标头，则响应仍会返回 200 OK。但是，只有允许的标头（例如 x-amz-meta-custom-header）才会返回并显示在响应中。

```
"ExposeHeaders":  [
    "x-amz-meta-custom-header1"
]
```

为确保所有标头都出现在响应中，请将所有允许的标头添加到 CORS 配置中的 ExposeHeaders 元素，如下所示。

```
"ExposeHeaders":  [
    "x-amz-meta-custom-header1",
    "x-amz-meta-custom-header2"
]
```

有关 S3 代理集成的 CORS 的注意事项

如果您遇到错误且已经检查了 S3 存储桶上的 CORS 配置，并且跨源请求已发送到 AWS CloudFront 等代理，请尝试以下操作：

- 将设置配置为支持 HTTP 请求使用 OPTIONS 方法。
- 将代理配置为转发以下标头：Origin、Access-Control-Request-Headers 和 Access-Control-Request-Method。

一些代理为 CORS 请求提供预定义的功能。例如，在 CloudFront 中，您可以配置一个策略，其中包括当源是 S3 存储桶时启用 CORS 请求的标头。有关更多信息，请参阅《CloudFront 开发人员指南》中的使用策略来控制源请求和使用托管式源请求策略。

使用 Amazon S3 进行开发

本部分介绍了与开发人员相关的使用 Amazon S3 的主题。有关更多信息，请查看以下主题。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅什么是 S3 Express One Zone？和目录桶。

主题

提出请求

Amazon S3 是一项 REST 服务。可以使用 REST API 或打包底层 Amazon S3 REST API 以简化编程任务的 AWS 开发工具包（请参阅 示例代码和库）包装程序库，向 Amazon S3 发送请求。

与 Amazon S3 的每一次交互都是经身份验证的或匿名的。身份验证是对尝试访问 (Amazon Web Services) (AWS) 产品的请求者身份进行验证的过程。经身份验证的请求必须包含可验证请求发送者的签名值。签名值的一部分是从请求者 AWS 的访问密钥（访问密钥 ID 和秘密访问密钥）生成的。有关获取访问密钥的更多信息，请参阅《AWS 一般参考》中的如何获取安全凭证？

如果您使用 AWS 开发工具包，则库将通过您提供的密钥计算签名。然而，如果您在应用程序中直接调用 REST API，您必须编写代码来计算签名并将它添加到请求中。

主题

- 通过 IPv6 向 Amazon S3 发出请求
- 使用 AWS 开发工具包提出请求
- 使用 REST API 提出请求

关于访问密钥

下面各部分将回顾您可以用于进行经身份验证的请求的访问密钥类型。

AWS 账户 访问密钥

账户访问密钥提供对账户拥有的 AWS 资源的完全访问权限。以下是访问密钥示例：

- 访问密钥 ID (20 个字符的字母数字字符串)。例如：AKIAIOSFODNN7EXAMPLE
- 秘密访问密钥 (40 个字符的字符串)。例如：wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

访问密钥 ID 可唯一标识 AWS 账户。可以使用这些访问密钥向 Amazon S3 发送经身份验证的请求。

IAM 用户访问密钥

您可以为公司创建一个 AWS 账户；但组织中可能有多个员工需要访问组织的 AWS 资源。共享您的 AWS 账户 访问密钥会降低安全性，但为每个员工创建单独的 AWS 账户 可能也不太实际。此外，您也可能无法轻松地共享资源 (例如，存储桶和对象)，因为它们由不同账户所拥有。要共享资源，您必须授予许可，这意味着需要额外的工作。

在这些情况中，您可以使用 AWS Identity and Access Management（IAM）在您的 AWS 账户 下创建具有其自己的访问密钥的用户，然后附加 IAM 用户策略以授予这些用户适当的资源访问权限。为了更好地管理这些用户，IAM 允许您创建用户组并授予适用于该组中所有用户的组级权限。

这些用户被称为 IAM 用户，您可以在内创建和管理这些用户AWS 父账户将控制用户访问的权限AWS 父 AWS 账户 负责控制和支付 IAM 用户创建的任何资源。这些 IAM 用户可以使用他们自己的安全证书，向 Amazon S3 发送经身份验证的请求。有关在 AWS 账户 下创建和管理用户的更多信息，请参阅 AWS Identity and Access Management 产品详细信息页面。

临时安全凭证

除了创建具有自己的访问密钥的 IAM 用户之外，IAM 还允许您向任何 IAM 用户授予临时安全凭证（临时访问密钥和安全令牌），以便这些用户可以访问您的 AWS 服务和资源。您也可以在之外的系统中管

理用户AWS 这些用户称为联合身份用户。此外，用户还可以是您创建的能访问您的 AWS 资源的应用程序。

IAM 提供了 AWS Security Token Service API，供您用于请求临时安全证书。您可以使用 AWS STS API 或 AWS 开发工具包来请求这些凭证。API 将返回临时安全凭证 (访问密钥 ID 和秘密访问密钥) 和安全令牌。这些凭证仅在您请求它们时指定的持续时间内有效。使用访问密钥 ID 和私有密钥的方法与您在使用 AWS 账户 或 IAM 用户访问密钥发送请求时使用它们的方法相同。此外，您必须在每个请求中包含您发送至 Amazon S3 的令牌。

IAM 用户可以请求这些临时安全证书以供自己使用，也可以分发给联合用户或应用程序使用。为联合用户请求临时安全证书时，您必须提供用户名称和 IAM 策略（定义您想要与这些临时安全证书关联的许可）。联合用户可获取的许可不能超过请求临时证书的父 IAM 用户的许可。

您还可以使用这些临时安全证书向 Amazon S3 发出请求。API 库会使用这些凭证计算必要的签名值，以便对您的请求进行身份验证。如果您使用过期的证书发送请求，Amazon S3 会拒绝请求。

有关在您的 REST API 请求中使用临时安全凭证对请求进行签名的信息，请参阅 签署和对 REST 请求进行身份验证。有关使用 AWS 开发工具包发送请求的信息，请参阅 使用 AWS 开发工具包提出请求。

有关 IAM 对临时安全凭证的支持的更多信息，请参阅《IAM 用户指南》中的临时安全凭证。

为了提高安全性，您可以通过配置存储桶策略，要求在访问您的 Amazon S3 资源时进行多重验证 (MFA)。有关信息，请参阅需要 MFA。在要求进行 MFA 才能访问您的 Amazon S3 资源之后，访问这些资源的唯一方式是提供使用 MFA 密钥创建的临时凭证。有关更多信息，请参阅 AWS Multi-Factor Authentication 详细信息页面和《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

请求终端节点

您可以向服务的预定义终端节点发送 REST 请求。有关所有 AWS 服务及其相应端点的列表，请参阅《AWS 一般参考》中的区域和端点。

通过 IPv6 向 Amazon S3 发出请求

除了 IPv4 协议之外，Amazon Simple Storage Service (Amazon S3) 还支持使用 Internet 协议版本 6 (IPv6) 访问 S3 存储桶的功能。Amazon S3 双堆栈终端节点支持通过 IPv6 和 IPv4 向 S3 存储桶发出请求。通过 IPv6 访问 Amazon S3 不额外收费。有关定价的更多信息，请参阅 Amazon S3 定价。

主题

- 通过 IPv6 发出请求入门

- 在 IAM 策略中使用 IPv6 地址
- 测试 IP 地址兼容性
- 使用 Amazon S3 双堆栈端点

通过 IPv6 发出请求入门

要通过 IPv6 向 S3 存储桶发出请求，您需要使用双堆栈终端节点。下一节介绍如何使用双堆栈终端节点通过 IPv6 发出请求。

下面是在尝试通过 IPv6 访问存储桶之前应了解的部分事项：

- 访问存储桶的客户端和网络必须支持使用 IPv6。
- 虚拟托管类型和路径类型请求必须都受支持，以便进行 IPv6 访问。有关更多信息，请参阅 Amazon S3 双堆栈端点。
- 如果您 AWS Identity and Access Management (IAM) 用户或存储桶策略中使用源 IP 地址筛选，则需要更新策略以包括 IPv6 地址范围。有关更多信息，请参阅 在 IAM 策略中使用 IPv6 地址。
- 如果使用 IPv6，服务器访问日志文件以 IPv6 格式输出 IP 地址。您需要对用于分析 Amazon S3 日志文件的现有工具、脚本和软件进行更新，以便它们能够分析 IPv6 格式的 `Remote IP` 地址。有关更多信息，请参阅 Amazon S3 服务器访问日志格式 和 使用服务器访问日志记录来记录请求。

> **Note**
>
> 如果在日志文件中遇到与 IPv6 地址相关的问题，请联系 AWS Support。

使用双堆栈终端节点通过 IPv6 发出请求

您可以使用双堆栈终端节点和 Amazon S3 API 调用通过 IPv6 发出请求。无论您是通过 IPv6 还是通过 IPv4 访问 Amazon S3，Amazon S3 API 操作的工作方式都相同。性能也应该是相同的。

如果使用 REST API，您是直接访问双堆栈终端节点。有关更多信息，请参阅 双堆栈端点。

如果使用 AWS Command Line Interface (AWS CLI) 和 AWS 开发工具包，可使用参数或标志以更改为双堆栈终端节点。您还可以在配置文件中直接将双堆栈终端节点指定为覆盖 Amazon S3 终端节点。

您可以通过以下任一方式，使用双堆栈终端节点通过 IPv6 访问存储桶：

- AWS CLI，请参阅从 AWS CLI 使用双堆栈端点。

- 有关 AWS 开发工具包，请参阅 从 AWS SDK 使用双堆栈端点。
- REST API，请参阅通过使用 REST API 向双堆栈终端节点发出请求。

不可通过 IPv6 使用的功能

当通过来自 S3 存储桶的 IPv6: Static 网站托管访问 S3 存储桶时，以下功能目前不受支持。

在 IAM 策略中使用 IPv6 地址

在尝试使用 IPv6 访问存储桶之前，您必须确保用于 IP 地址筛选的所有 IAM 用户或 S3 存储桶策略已更新为包括 IPv6 地址范围。在开始使用 IPv6 时，未更新为处理 IPv6 地址的 IP 地址筛选策略可能导致客户端错误地丢失或获得存储桶访问权。有关使用 IAM 管理访问权限的更多信息，请参阅 Amazon S3 的身份和访问管理。

筛选 IP 地址的 IAM 策略使用 IP 地址条件运算符。下面的存储桶策略通过使用 IP 地址条件运算符确定允许的 IPv4 地址范围为 54.240.143.*。此范围之外的所有 IP 地址对存储桶 (`examplebucket`) 的访问都会被拒绝。由于所有 IPv6 地址都在允许范围之外，此策略会阻止 IPv6 地址访问 `examplebucket`。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

您可以将存储桶策略的 `Condition` 元素修改为同时允许 IPv4 (`54.240.143.0/24`) 和 IPv6 (`2001:DB8:1234:5678::/64`) 地址范围，如以下示例所示。您可以使用示例中所示的相同类型的 `Condition` 块来更新 IAM 用户和存储桶策略。

```
       "Condition": {
```

```
         "IpAddress": {
            "aws:SourceIp": [
              "54.240.143.0/24",
               "2001:DB8:1234:5678::/64"
             ]
          }
        }
```

在使用 IPv6 之前，必须对使用 IP 地址筛选的所有相关 IAM 用户和存储桶策略进行更新。我们建议不要在存储桶策略中使用 IP 地址筛选。

您可以在 https://console.aws.amazon.com/iam/ 上使用 IAM 控制台查看 IAM 用户策略。有关 IAM 的更多信息，请参阅 IAM 用户指南。有关编辑 S3 存储桶策略的信息，请参阅使用 Amazon S3 控制台添加存储桶策略。

测试 IP 地址兼容性

如果使用 Linux/Unix 或 Mac OS X，则通过使用下例所示的 `curl` 命令，可以测试您能否通过 IPv6 访问双堆栈终端节点：

Example

```
curl -v  http://s3.dualstack.us-west-2.amazonaws.com/
```

您获得的信息如下例所示。如果通过 IPv6 连接，则连接的 IP 地址将会是 IPv6 地址。

```
* About to connect() to s3-us-west-2.amazonaws.com port 80 (#0)
*   Trying IPv6 address... connected
* Connected to s3.dualstack.us-west-2.amazonaws.com (IPv6 address) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1 OpenSSL/1.0.1t
 zlib/1.2.3
> Host: s3.dualstack.us-west-2.amazonaws.com
```

如果使用 Microsoft Windows 7 或 Windows 10，则通过使用下例所示的 `ping` 命令，可以测试您能否通过 IPv6 或 IPv4 访问双堆栈终端节点。

```
ping ipv6.s3.dualstack.us-west-2.amazonaws.com
```

使用 Amazon S3 双堆栈端点

Amazon S3 双堆栈端点支持通过 IPv6 和 IPv4 向 S3 存储桶发出请求。本节介绍如何使用双堆栈端点。

主题

Amazon S3 双堆栈端点

当您向双堆栈端点发出请求时，存储桶 URL 解析为 IPv6 或 IPv4 地址。有关如何通过 IPv6 访问存储桶的更多信息，请参阅通过 IPv6 向 Amazon S3 发出请求。

当使用 REST API 时，您通过使用端点名称（URI）直接访问 Amazon S3 端点。通过使用虚拟托管类型或路径类型端点名称，您可以通过双堆栈端点访问 S3 存储桶。Amazon S3 只支持区域双堆栈端点名称，这意味着，您必须在名称中指定区域。

双堆栈虚拟托管类型和路径类型端点名称使用以下命名惯例：

- 虚拟托管型双堆栈端点：

  bucketname.s3.dualstack.*aws-region*.amazonaws.com

- 路径型双堆栈端点：

  s3.dualstack.*aws-region*.amazonaws.com/*bucketname*

有关端点名称样式的更多信息，请参阅访问和列出 Amazon S3 存储桶。有关 Amazon S3 端点的列表，请参阅《AWS 一般参考》中的区域和端点。

> **Important**
>
> 您可对双堆栈端点使用传输加速。有关更多信息，请参阅 开始使用 Amazon S3 Transfer Acceleration。

> **Note**
>
> 访问 Amazon S3 的两种类型的 VPC 端点（接口 VPC 端点和网关 VPC 端点）不支持双堆栈。有关 Amazon S3 的 VPC 端点的更多信息，请参阅AWS PrivateLink：表示 Amazon S3。

如果使用 AWS Command Line Interface（AWS CLI）和 AWS SDK，可使用参数或标志以更改为双堆栈端点。您还可以在配置文件中直接将双堆栈端点指定为覆盖 Amazon S3 端点。下面几节介绍如何从 AWS CLI 和 AWS SDK 使用双堆栈端点。

从 AWS CLI 使用双堆栈端点

本节提供用于向双堆栈端点发出请求的 AWS CLI 命令示例。有关设置 AWS CLI 的说明，请参阅使用 AWS CLI 进行 Amazon S3 开发。

在 AWS Config 文件中的配置文件中将配置值 `use_dualstack_endpoint` 设置为 `true`，使 `s3` 和 `s3api` AWS CLI 命令发出的所有 Amazon S3 请求定向到指定区域的双堆栈端点。您可以在配置文件或命令中使用 `--region` 选项指定区域。

通过 AWS CLI 使用双堆栈端点时，支持 `path` 和 `virtual` 寻址类型。配置文件中设置的寻址类型控制着主机名或 URL 中是否包含存储桶名称。默认情况下，CLI 尝试使用虚拟类型，但是如果需要，会改为使用路径类型。有关更多信息，请参阅 AWS CLI Amazon S3 配置。

您也可以使用命令进行配置更改，如下例所示，在默认配置文件中将 `use_dualstack_endpoint` 设置为 `true`，将 `addressing_style` 设置为 `virtual`。

```
$ aws configure set default.s3.use_dualstack_endpoint true
$ aws configure set default.s3.addressing_style virtual
```

如果需要只对指定的 AWS CLI 命令（并非所有命令）使用双堆栈端点，可以使用以下任一方法：

- 您可以通过将任何 `--endpoint-url` 或 `https://s3.dualstack.aws-region.amazonaws.com` 命令的 `http://s3.dualstack.aws-region.amazonaws.com` 参数设置为 `s3` 或 `s3api` 来对每条命令使用双堆栈端点。

  ```
  $ aws s3api list-objects --bucket bucketname --endpoint-url https://s3.dualstack.aws-region.amazonaws.com
  ```

- 您可以在 AWS Config 文件中设置单独的配置文件。例如，创建一个将 `use_dualstack_endpoint` 设置为 `true` 的配置文件和一个不设置 `use_dualstack_endpoint` 的配置文件。在运行命令时，根据是否需要使用双堆栈端点来指定要使用的配置文件。

> **Note**
>
> 当使用 AWS CLI 时，当前不能对双堆栈端点使用传输加速。但是，即将推出对 AWS CLI 的支持。有关更多信息，请参阅 使用 AWS CLI。

从 AWS SDK 使用双堆栈端点

本节提供一些示例，介绍如何使用 AWS SDK 来访问双堆栈端点。

AWS SDK for Java双堆栈端点示例

以下示例演示如何使用 AWS SDK for Java 在创建 Amazon S3 客户端时启用双堆栈端点。

有关创建和测试有效 Java 示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class DualStackEndpoints {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            // Create an Amazon S3 client with dual-stack endpoints enabled.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .withDualstackEnabled(true)
                    .build();
```

```
            s3Client.listObjects(bucketName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

如果要在 Windows 上使用AWS SDK for Java，可能必须设置以下 Java 虚拟机 (JVM) 属性：

```
java.net.preferIPv6Addresses=true
```

AWS .NET SDK 双堆栈端点示例

在使用 AWS SDK for .NET 时，请使用 `AmazonS3Config` 类来启用双堆栈端点，如下例所示。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DualStackEndpointTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            var config = new AmazonS3Config
            {
                UseDualstackEndpoint = true,
                RegionEndpoint = bucketRegion
```

```
            };
            client = new AmazonS3Client(config);
            Console.WriteLine("Listing objects stored in a bucket");
            ListingObjectsAsync().Wait();
        }

        private static async Task ListingObjectsAsync()
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 10
                };
                ListObjectsV2Response response;
                do
                {
                    response = await client.ListObjectsV2Async(request);

                    // Process the response.
                    foreach (S3Object entry in response.S3Objects)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }
                    Console.WriteLine("Next Continuation Token: {0}",
 response.NextContinuationToken);
                    request.ContinuationToken = response.NextContinuationToken;
                } while (response.IsTruncated == true);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

有关所列对象的完整 .NET 示例，请参阅以编程方式列出对象键。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

从 REST API 使用双堆栈端点

有关如何使用 REST API 向双堆栈端点发出请求的信息，请参阅通过使用 REST API 向双堆栈终端节点发出请求。

使用 AWS 开发工具包提出请求

主题

您可以使用 AWS 开发工具包或通过直接在应用程序中进行 REST API 调用，将经身份验证的请求发送到 Amazon S3。AWS 开发工具包 API 使用您提供的凭证来计算用于身份验证的签名。若您直接在应用程序中使用 REST API，您必须编写所需的代码来计算用于对您的请求进行身份验证的签名。有关可用 AWS 开发工具包的列表，请转到 示例代码和库。

使用 AWS 账户 或 IAM 用户凭证发出请求

您可以使用 AWS 账户 或 IAM 用户安全凭证向 Amazon S3 发送经身份验证的请求。本部分提供的示例演示了如何使用 AWS SDK for Java、AWS SDK for .NET 和 AWS SDK for PHP 发送经身份验证的请求。有关 AWS 可用开发工具包的列表，请转到 示例代码和库。

每个 AWS 开发工具包都使用开发工具包特有的凭证提供程序链来查找和使用凭证，并代表凭证拥有者执行操作。所有这些凭证提供程序链的共同点是，它们都会寻找您的本地 AWS 凭证文件。

有关更多信息，请参阅以下主题：

主题

创建本地 AWS 凭证文件

为 AWS 开发工具包配置凭证的最简单方法是使用 AWS 凭证文件。如果您使用 AWS Command Line Interface (AWS CLI)，那么您可能已经配置了本地 AWS 凭证文件。否则，请按照以下步骤设置凭证文件：

1. 登录 AWS Management Console，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

2. 创建一个新用户，其权限仅限于您希望您的代码有权访问的服务和操作。有关创建新用户的更多信息，请参阅创建 IAM 用户（控制台），并按照直至步骤 8 的说明进行操作。
3. 选择 Download .csv（下载 .csv）以保存 AWS 凭证的本地副本。
4. 在您的计算机上，导航至主目录，并创建 `.aws` 目录。在基于 Unix 的系统 (例如 Linux 或 OS X) 上，它在以下位置：

   ```
   ~/.aws
   ```

   在 Windows 上，它在以下位置：

   ```
   %HOMEPATH%\.aws
   ```
5. 在 `.aws` 目录中，创建名为 `credentials` 的新文件。
6. 打开您从 IAM 控制台中下载的凭证 `.csv` 文件，并使用以下格式将其内容复制到 `credentials` 文件：

   ```
   [default]
   aws_access_key_id = your_access_key_id
   aws_secret_access_key = your_secret_access_key
   ```
7. 保存 `credentials` 文件，并删除在步骤 3 中下载的 `.csv` 文件。

您的共享凭证文件现在已在本地计算机上配置完毕，可以与 AWS 开发工具包一起使用。

使用 AWS 开发工具包发送经身份验证的请求

使用 AWS 开发工具包发送经过身份验证的请求。有关发送经过身份验证的请求的更多信息，请参阅 AWS 安全证书或 IAM Identity Center 身份验证。

Java

要使用 AWS 账户 或 IAM 用户凭证向 Amazon S3 发送经身份验证的请求，请执行以下操作：

- 使用 `AmazonS3ClientBuilder` 类创建 `AmazonS3Client` 实例。
- 运行 `AmazonS3Client` 方法之一，以向 Amazon S3 发送请求。客户端将通过您提供的凭证生成所需的签名并将其包含在请求中。

以下示例将执行上述任务。有关创建和测试有效示例的信息，请参阅《AWS SDK for Java 开发人员指南》中的入门。

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsRequest;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.S3ObjectSummary;

import java.io.IOException;
import java.util.List;

public class MakingRequests {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Get a list of objects in the bucket, two at a time, and
            // print the name and size of each object.
            ListObjectsRequest listRequest = new
 ListObjectsRequest().withBucketName(bucketName).withMaxKeys(2);
            ObjectListing objects = s3Client.listObjects(listRequest);
            while (true) {
                List<S3ObjectSummary> summaries = objects.getObjectSummaries();
                for (S3ObjectSummary summary : summaries) {
                    System.out.printf("Object \"%s\" retrieved with size %d\n",
 summary.getKey(), summary.getSize());
                }
                if (objects.isTruncated()) {
                    objects = s3Client.listNextBatchOfObjects(objects);
```

```
                } else {
                    break;
                }
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

使用 AWS 账户 或 IAM 用户凭证发送经身份验证的请求：

- 创建 AmazonS3Client 类的实例。
- 运行 AmazonS3Client 方法之一，以向 Amazon S3 发送请求。客户端将通过您提供的凭证生成所需的签名并将其包含在它发送到 Amazon S3 的请求中。

有关更多信息，请参阅 使用 AWS 账户 或 IAM 用户凭证发出请求。

> **Note**
>
> - 您可以在不提供安全凭证的情况下创建 AmazonS3Client 客户端。使用此客户端发送的请求是匿名的请求，它们不带签名。如果为不是公开可用的资源发送了匿名请求，Amazon S3 将返回错误。
> - 您可以创建 AWS 账户 并创建所需的用户。您还可以管理这些用户的凭证。您需要这些凭证才能执行以下示例中的任务。有关更多信息，请参阅《AWS SDK for .NET 开发人员指南》中的配置 AWS 凭证。
>
>   然后，您还可以将应用程序配置为主动检索配置文件和凭证，然后在创建 AWS 服务客户端时明确使用这些凭证。有关更多信息，请参阅《AWS SDK for .NET 开发人员指南》中的 访问应用程序中的凭证和配置文件。

以下 C# 示例说明如何执行上述任务。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class MakeS3RequestTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            using (client = new AmazonS3Client(bucketRegion))
            {
                Console.WriteLine("Listing objects stored in a bucket");
                ListingObjectsAsync().Wait();
            }
        }

        static async Task ListingObjectsAsync()
        {
            try
            {
                ListObjectsRequest request = new ListObjectsRequest
                {
                    BucketName = bucketName,
                    MaxKeys = 2
                };
                do
                {
                    ListObjectsResponse response = await
 client.ListObjectsAsync(request);
                    // Process the response.
```

```
                    foreach (S3Object entry in response.S3Objects)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }

                    // If the response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.Marker = response.NextMarker;
                    }
                    else
                    {
                        request = null;
                    }
                } while (request != null);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

有关有效示例，请参阅Amazon S3 对象概述和存储桶概述。您可以使用 AWS 账户 或 IAM 用户证书测试这些示例。

例如，要列出您的存储桶中的所有对象键，请参阅 以编程方式列出对象键。

PHP

此部分说明如何使用版本 3 的 AWS SDK for PHP 中的类来通过 AWS 账户 或 IAM 用户凭证发送经身份验证的请求。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

以下 PHP 示例说明客户端如何使用您的安全凭证发出请求以列出您的账户的所有存储桶。

Example

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
]);

// Retrieve the list of buckets.
$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // Print the list of objects to the page.
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

> **Note**
>
> 您可以在不提供安全凭证的情况下创建 `S3Client` 客户端。使用此客户端发送的请求是匿名的请求，它们不带签名。如果为不是公开可用的资源发送了匿名请求，Amazon S3 将返回错误。有关更多信息，请参阅 AWS SDK for PHP 文档 中的 创建匿名客户端。

有关有效示例，请参阅Amazon S3 对象概述。您可以使用 AWS 账户 或 IAM 用户凭证测试这些示例。

有关列出存储桶中的对象键的示例，请参阅以编程方式列出对象键。

Ruby

使用 AWS SDK for Ruby 的版本 3 调用 Amazon S3 之前，必须设置 AWS 访问凭证以便开发工具包用来验证您对存储桶和对象的访问权限。如果您在本地系统上的 AWS 凭证配置文件中设置了共享凭证，则适用于 Ruby 的开发工具包的版本 3 可以在无需您在代码中声明这些凭证的情况下使用它们。有关设置共享凭证的更多信息，请参阅使用 AWS 账户 或 IAM 用户凭证发出请求。

以下 Ruby 代码段使用本地计算机上凭证文件中的共享 AWS 凭证，对要获取特定存储桶中的所有对象键名的请求进行身份验证。它将执行以下操作：

1. 创建 `Aws::S3::Client` 类的实例。
2. 使用 `list_objects_v2` 的 `Aws::S3::Client` 方法，通过枚举存储桶中的对象向 Amazon S3 发出请求。客户端将根据计算机上 AWS 凭证文件中的凭证生成所需的签名值，并将其包含在发送给 Amazon S3 的请求中。
3. 将对象键名阵列打印到终端。

Example

```
# Prerequisites:
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'doc-example-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )
```

```
  if objects.count.positive?
    puts "The object keys in this bucket are (first 50 objects):"
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts "No objects found in this bucket."
  end

  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  return false
end

# Example usage:
def run_me
  region = "us-west-2"
  bucket_name = "BUCKET_NAME"
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

即使没有本地 AWS 凭证文件，您仍可以创建 `Aws::S3::Client` 资源并针对 Amazon S3 存储桶和对象运行代码。使用适用于 Ruby 的开发工具包的版本 3 发送的请求是匿名的，默认情况下没有签名。如果为不是公开可用的资源发送了匿名请求，Amazon S3 将返回错误。

可以针对适用于 Ruby 的开发工具包应用程序使用并扩展前一个代码段，如下面的更可靠的示例所示。

```
# Prerequisites:
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
```

```
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'doc-example-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts "The object keys in this bucket are (first 50 objects):"
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts "No objects found in this bucket."
  end

  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  return false
end

# Example usage:
def run_me
  region = "us-west-2"
  bucket_name = "BUCKET_NAME"
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

Go

Example

以下示例使用由 SDK for Go 从共享凭证文件中自动加载的 AWS 凭证。

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

相关资源

- 使用 AWS SDK 通过 Amazon S3 进行开发
- 面向 Amazon S3 Aws\S3\S3Client 类的 AWS SDK for PHP
- AWS SDK for PHP 文档

使用 IAM 用户临时凭证发出请求

AWS 账户 或 IAM 用户可以请求临时安全证书，然后使用它们来向 Amazon S3 发送经身份验证的请求。本节提供了一些示例，介绍了如何使用 AWS SDK for Java、.NET 和 PHP 获取临时安全证书，以及如何使用这些证书来对您发送至 Amazon S3 的请求进行身份验证。

Java

IAM 用户或 AWS 账户 可以使用采用 AWS SDK for Java 的临时安全凭证（请参阅 提出请求），然后使用这些凭证访问 Amazon S3。在指定会话持续时间结束后，这些凭证将过期。

默认情况下，会话的持续时间为一个小时。如果您使用了 IAM 用户凭证，则可在请求临时安全凭证时指定持续时间（15 分钟到角色的最长会话持续时间）。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的临时安全凭证。有关发出请求的更多信息，请参阅提出请求。

获取临时安全凭证并访问 Amazon S3

1. 创建 `AWSSecurityTokenService` 类的实例。有关提供凭证的信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。
2. 通过调用安全令牌服务 (STS) 客户端的 `assumeRole()` 方法，检索所需角色的临时安全凭证。
3. 将临时安全凭证打包到 `BasicSessionCredentials` 对象中。您可以使用此对象来向您的 Amazon S3 客户端提供临时安全凭证。
4. 使用临时安全凭证创建 `AmazonS3Client` 类的实例。您可以使用此客户端向 Amazon S3 发送请求。如果使用过期凭证发送请求，Amazon S3 将返回错误。

> **Note**
>
> 如果使用 AWS 账户 安全凭证获取临时安全凭证，则临时凭证的有效期只有一小时。只有当您使用 IAM 用户证书来请求会话时，您才可以指定会话持续时间。

以下示例列出了指定存储桶中的一组对象键。该示例将为会话获取临时安全凭证，然后使用这些凭证向 Amazon S3 发送经身份验证的请求。

如果要使用 IAM 用户凭证测试示例，则必须在 AWS 账户 下创建一个 IAM 用户。有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的创建您的第一个 IAM 用户和管理员组。

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.securitytoken.model.Credentials;

public class MakingRequestsWithIAMTempCredentials {
    public static void main(String[] args) {
        String clientRegion = "*** Client region ***";
        String roleARN = "*** ARN for role to be assumed ***";
        String roleSessionName = "*** Role session name ***";
        String bucketName = "*** Bucket name ***";

        try {
            // Creating the STS client is part of your trusted code. It has
            // the security credentials you use to obtain temporary security
 credentials.
            AWSSecurityTokenService stsClient =
 AWSSecurityTokenServiceClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Obtain credentials for the IAM role. Note that you cannot assume the
 role of
            // an AWS root account;
            // Amazon S3 will deny access. You must use credentials for an IAM user
 or an
            // IAM role.
            AssumeRoleRequest roleRequest = new AssumeRoleRequest()
                    .withRoleArn(roleARN)
                    .withRoleSessionName(roleSessionName);
            AssumeRoleResult roleResponse = stsClient.assumeRole(roleRequest);
            Credentials sessionCredentials = roleResponse.getCredentials();
```

```
            // Create a BasicSessionCredentials object that contains the credentials
 you
            // just retrieved.
            BasicSessionCredentials awsCredentials = new BasicSessionCredentials(
                    sessionCredentials.getAccessKeyId(),
                    sessionCredentials.getSecretAccessKey(),
                    sessionCredentials.getSessionToken());

            // Provide temporary security credentials so that the Amazon S3 client
            // can send authenticated requests to Amazon S3. You create the client
            // using the sessionCredentials object.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new
 AWSStaticCredentialsProvider(awsCredentials))
                    .withRegion(clientRegion)
                    .build();

            // Verify that assuming the role worked and the permissions are set
 correctly
            // by getting a set of object keys from the bucket.
            ObjectListing objects = s3Client.listObjects(bucketName);
            System.out.println("No. of Objects: " +
 objects.getObjectSummaries().size());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

IAM 用户或 AWS 账户 可以使用 AWS SDK for .NET 请求临时安全凭证，然后使用这些凭证来访问 Amazon S3。在会话持续时间结束后，这些凭证将过期。

默认情况下，会话的持续时间为一个小时。如果您使用了 IAM 用户凭证，则可在请求临时安全凭证时指定持续时间（15 分钟到角色的最长会话持续时间）。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的临时安全凭证。有关发出请求的更多信息，请参阅提出请求。

获取临时安全凭证并访问 Amazon S3

1. 创建AWS Security Token Service客户端的实例 `AmazonSecurityTokenServiceClient`。有关提供凭证的信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。
2. 通过调用您在上一步骤中创建的 STS 客户端的 `GetSessionToken` 方法，开始会话。您可以使用 `GetSessionTokenRequest` 对象向此方法提供会话信息。

   此方法将返回您的临时安全凭证。
3. 将临时安全凭证打包在 `SessionAWSCredentials` 对象的实例中。您可以使用此对象来向您的 Amazon S3 客户端提供临时安全凭证。
4. 通过传入临时安全凭证创建 `AmazonS3Client` 类的实例。您可以使用此客户端向 Amazon S3 发送请求。如果您使用过期的凭证发送请求，Amazon S3 将返回错误。

> **Note**
>
> 如果使用 AWS 账户 安全凭证获取临时安全凭证，则这些凭证仅在一小时内有效。仅当使用 IAM 用户凭证请求会话时，才能指定会话持续时间。

以下 C# 示例列出了指定存储桶中的对象键。为了展示这个过程，示例会为默认一小时的会话获取临时安全凭证，然后使用这些凭证来向 Amazon S3 发送经身份验证的请求。

如果要使用 IAM 用户凭证测试示例，则必须在 AWS 账户 下创建一个 IAM 用户。有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的创建您的第一个 IAM 用户和管理员组。有关发出请求的更多信息，请参阅提出请求。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
```

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempCredExplicitSessionStartTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                // Credentials use the default AWS SDK for .NET credential search
 chain.
                // On local development machines, this is your default profile.
                Console.WriteLine("Listing objects stored in a bucket");
                SessionAWSCredentials tempCredentials = await
 GetTemporaryCredentialsAsync();

                // Create a client by providing temporary security credentials.
                using (s3Client = new AmazonS3Client(tempCredentials, bucketRegion))
                {
                    var listObjectRequest = new ListObjectsRequest
                    {
                        BucketName = bucketName
                    };
                    // Send request to Amazon S3.
                    ListObjectsResponse response = await
 s3Client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);
                }
            }
            catch (AmazonS3Exception s3Exception)
```

```
            {
                Console.WriteLine(s3Exception.Message, s3Exception.InnerException);
            }
            catch (AmazonSecurityTokenServiceException stsException)
            {
                Console.WriteLine(stsException.Message,
stsException.InnerException);
            }
        }

        private static async Task<SessionAWSCredentials>
GetTemporaryCredentialsAsync()
        {
            using (var stsClient = new AmazonSecurityTokenServiceClient())
            {
                var getSessionTokenRequest = new GetSessionTokenRequest
                {
                    DurationSeconds = 7200 // seconds
                };

                GetSessionTokenResponse sessionTokenResponse =
                            await
stsClient.GetSessionTokenAsync(getSessionTokenRequest);

                Credentials credentials = sessionTokenResponse.Credentials;

                var sessionCredentials =
                    new SessionAWSCredentials(credentials.AccessKeyId,
                                              credentials.SecretAccessKey,
                                              credentials.SessionToken);
                return sessionCredentials;
            }
        }
    }
}
```

PHP

有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

IAM 用户或 AWS 账户 可使用 AWS SDK for PHP 的版本 3 请求临时安全凭证。之后，它可使用临时凭证访问 Amazon S3。这些凭证将在会话持续时间结束时到期。

默认情况下，会话的持续时间为一个小时。如果您使用了 IAM 用户凭证，则可在请求临时安全凭证时指定持续时间（15 分钟到角色的最长会话持续时间）。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的临时安全凭证。有关发出请求的更多信息，请参阅提出请求。

> **Note**
>
> 如果您使用AWS 账户 安全凭证获取临时安全凭证，则临时安全凭证的有效期仅为一个小时。只有当您使用 IAM 用户证书来请求会话时，您才可以指定会话持续时间。

Example

以下 PHP 示例使用临时安全凭证列出指定存储桶中的对象键。该示例将为默认一小时的会话获取临时安全凭证，然后使用这些凭证向 Amazon S3 发送经身份验证的请求。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

如果要使用 IAM 用户凭证测试示例，则必须在 AWS 账户 下创建一个 IAM 用户。有关如何创建 IAM 用户的信息，请参阅《IAM 用户指南》中的创建您的第一个 IAM 用户和管理员组。有关在使用 IAM 用户凭证请求会话时设置会话持续时间的示例，请参阅使用 IAM 用户临时凭证发出请求 。

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

$sessionToken = $sts->getSessionToken();

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key'    => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token'  => $sessionToken['Credentials']['SessionToken']
```

```
    ]
]);

$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // List objects
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

IAM 用户或 AWS 账户 可以使用 AWS SDK for Ruby 请求临时安全凭证，然后使用这些凭证来访问 Amazon S3。在会话持续时间结束后，这些凭证将过期。

默认情况下，会话的持续时间为一个小时。如果您使用了 IAM 用户凭证，则可在请求临时安全凭证时指定持续时间（15 分钟到角色的最长会话持续时间）。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的临时安全凭证。有关发出请求的更多信息，请参阅提出请求。

> **Note**
>
> 如果您使用AWS 账户 安全凭证获取临时安全凭证，则临时安全凭证的有效期仅为一个小时。只有当您使用 IAM 用户证书来请求会话时，您才可以指定会话持续时间。

以下 Ruby 示例将创建一个临时用户来列出指定存储桶中的项目 1 小时。要使用此示例，则必须具有 AWS 凭证，此类凭证具有创建新的 AWS Security Token Service (AWS STS) 客户端和列出 Amazon S3 存储桶所需的权限。

```
# Prerequisites:
# - A user in AWS Identity and Access Management (IAM). This user must
#   be able to assume the following IAM role. You must run this code example
#   within the context of this user.
# - An existing role in IAM that allows all of the Amazon S3 actions for all of the
#   resources in this code example. This role must also trust the preceding IAM
 user.
# - An existing S3 bucket.

require "aws-sdk-core"
require "aws-sdk-s3"
require "aws-sdk-iam"

# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return true if response.user.user_name
rescue Aws::IAM::Errors::NoSuchEntity
  # User doesn't exist.
rescue StandardError => e
  puts "Error while determining whether the user " \
    "'#{user_name}' exists: #{e.message}"
end

# Creates a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS:IAM::Types::User] The new user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = create_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def create_user(iam_client, user_name)
  response = iam_client.create_user(user_name: user_name)
  return response.user
rescue StandardError => e
```

```
  puts "Error while creating the user '#{user_name}': #{e.message}"
end

# Gets a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS:IAM::Types::User] The existing user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def get_user(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return response.user
rescue StandardError => e
  puts "Error while getting the user '#{user_name}': #{e.message}"
end

# Checks whether a role exists in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The role's name.
# @return [Boolean] true if the role exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless role_exists?(iam_client, 'my-role')
def role_exists?(iam_client, role_name)
  response = iam_client.get_role(role_name: role_name)
  return true if response.role.role_name
rescue StandardError => e
  puts "Error while determining whether the role " \
    "'#{role_name}' exists: #{e.message}"
end

# Gets credentials for a role in IAM.
#
# @param sts_client [Aws::STS::Client] An initialized AWS STS client.
# @param role_arn [String] The role's Amazon Resource Name (ARN).
# @param role_session_name [String] A name for this role's session.
# @param duration_seconds [Integer] The number of seconds this session is valid.
# @return [AWS::AssumeRoleCredentials] The credentials.
# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')
```

```
#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: role_session_name,
    duration_seconds: duration_seconds
  )
rescue StandardError => e
  puts "Error while getting credentials: #{e.message}"
end

# Checks whether a bucket exists in Amazon S3.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The name of the bucket.
# @return [Boolean] true if the bucket exists; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless bucket_exists?(s3_client, 'doc-example-bucket')
def bucket_exists?(s3_client, bucket_name)
  response = s3_client.list_buckets
  response.buckets.each do |bucket|
    return true if bucket.name == bucket_name
  end
rescue StandardError => e
  puts "Error while checking whether the bucket '#{bucket_name}' " \
    "exists: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'doc-example-bucket')
```

```
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts "Name => ETag"
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end
```

相关资源

- 使用 AWS SDK 通过 Amazon S3 进行开发
- 面向 Amazon S3 Aws\S3\S3Client 类的 AWS SDK for PHP
- AWS SDK for PHP 文档

使用联合身份用户临时凭证提出请求

您可以请求临时安全凭证并将它们提供给需要访问您的 AWS 资源的联合身份用户或应用程序。本节提供了一些示例，介绍了如何使用 AWS 开发工具包为联合身份用户或应用程序获取临时安全证书以及如何使用这些证书向 Amazon S3 发送经身份验证的请求。有关可用 AWS 开发工具包的列表，请参阅 示例代码和库。

> **Note**
>
> AWS 账户 和 IAM 用户皆可请求适用于联合身份用户的临时安全证书。然而，为了提高安全性，只有具有所需许可的 IAM 用户才能请求这些临时证书，这样可确保联合用户能够从请求的 IAM 用户中获得最多的许可。在某些应用程序中，您可能会发现创建一个具有特定权限的 IAM 用户以专门用于向联合身份用户和应用程序授予临时安全凭证比较合适。

Java

您可以为联合身份用户和应用程序提供临时安全凭证，以便它们可以发送经身份验证的请求，从而访问您的 AWS 资源。请求这些临时凭证时，必须提供用户名和 IAM 策略（描述要授予的资源权限）。默认情况下，会话的持续时间为一个小时。您可以在为联合身份用户和应用程序请求临时安全凭证时，显式地设置其他持续时间值。

> **Note**
>
> 为了提高请求适用于联合身份用户和应用程序的临时安全凭证时的安全性，建议使用仅具有所需访问权限的专用 IAM 用户。您创建的临时用户可获取的许可绝对不能超过请求临时安全证书的 IAM 用户的许可。有关更多信息，请参阅 AWS Identity and Access Management 常见问题。

要提供安全凭证并发送经身份验证的请求以访问资源，请执行以下操作：

- 创建 `AWSSecurityTokenServiceClient` 类的实例。
- 通过调用安全令牌服务 (STS) 客户端的 `getFederationToken()` 方法启动会话。提供会话信息，包括要附加到临时凭证的用户名和 IAM 策略。您可以提供可选的会话持续时间。此方法将返回您的临时安全凭证。
- 将临时安全凭证打包在 `BasicSessionCredentials` 对象的实例中。您可以使用此对象来向您的 Amazon S3 客户端提供临时安全凭证。

- 使用临时安全凭证创建 AmazonS3Client 类的实例。您可以使用此客户端向 Amazon S3 发送请求。如果您使用过期的凭证发送请求，Amazon S3 将返回错误。

Example

该示例列出了指定 S3 存储桶中的键。在该示例中，您将为联合身份用户的时长两个小时的会话获取临时安全凭证，然后使用这些凭证向 Amazon S3 发送经身份验证的请求。要运行该示例，您需要创建具有附加策略（允许用户请求临时安全凭证和列出 AWS 资源）的 IAM 用户。以下策略将实现此操作：

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的创建您的第一个 IAM 用户和管理员组。

创建 IAM 用户并附加上述策略之后，您可以运行以下示例。有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

```
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

import java.io.IOException;

public class MakingRequestsWithFederatedTempCredentials {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Specify bucket name ***";
        String federatedUser = "*** Federated user name ***";
        String resourceARN = "arn:aws:s3:::" + bucketName;

        try {
            AWSSecurityTokenService stsClient = AWSSecurityTokenServiceClientBuilder
                    .standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            GetFederationTokenRequest getFederationTokenRequest = new
 GetFederationTokenRequest();
            getFederationTokenRequest.setDurationSeconds(7200);
            getFederationTokenRequest.setName(federatedUser);

            // Define the policy and add it to the request.
            Policy policy = new Policy();
            policy.withStatements(new Statement(Effect.Allow)
                    .withActions(S3Actions.ListObjects)
                    .withResources(new Resource(resourceARN)));
            getFederationTokenRequest.setPolicy(policy.toJson());

            // Get the temporary security credentials.
            GetFederationTokenResult federationTokenResult =
 stsClient.getFederationToken(getFederationTokenRequest);
            Credentials sessionCredentials = federationTokenResult.getCredentials();

            // Package the session credentials as a BasicSessionCredentials
            // object for an Amazon S3 client object to use.
```

```
            BasicSessionCredentials basicSessionCredentials = new
BasicSessionCredentials(
                    sessionCredentials.getAccessKeyId(),
                    sessionCredentials.getSecretAccessKey(),
                    sessionCredentials.getSessionToken());
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new
AWSStaticCredentialsProvider(basicSessionCredentials))
                    .withRegion(clientRegion)
                    .build();

            // To verify that the client works, send a listObjects request using
            // the temporary security credentials.
            ObjectListing objects = s3Client.listObjects(bucketName);
            System.out.println("No. of Objects = " +
objects.getObjectSummaries().size());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

您可以为联合身份用户和应用程序提供临时安全凭证，以便它们可以发送经身份验证的请求，从而访问您的 AWS 资源。请求这些临时凭证时，必须提供用户名和 IAM 策略（描述要授予的资源权限）。默认情况下，会话的持续时间为一个小时。您可以在为联合身份用户和应用程序请求临时安全凭证时，显式地设置其他持续时间值。有关发送经身份验证的请求的信息，请参阅提出请求。

> **Note**
>
> 当请求适用于联合身份用户和应用程序的临时安全凭证时，为了提高安全性，建议使用仅具有所需访问权限的专用 IAM 用户。您创建的临时用户可获取的许可绝对不能超过请求临时安全证书的 IAM 用户的许可。有关更多信息，请参阅 AWS Identity and Access Management 常见问题。

您应当执行以下操作：

- 创建 AWS Security Token Service客户端 `AmazonSecurityTokenServiceClient` 类的实例。
- 通过调用 STS 客户端的 `GetFederationToken` 方法开始会话。您需要提供会话信息，包括要附加到临时凭证的用户名和 IAM 策略。（可选）您可以提供会话持续时间。此方法将返回您的临时安全凭证。
- 将临时安全凭证打包在 `SessionAWSCredentials` 对象的实例中。您可以使用此对象来向您的 Amazon S3 客户端提供临时安全凭证。
- 通过传递临时安全凭证创建 `AmazonS3Client` 类的实例。您可使用此客户端向 Amazon S3 发送请求。如果您使用过期的凭证发送请求，Amazon S3 将返回错误。

Example

以下 C# 示例列出了指定存储桶中的密钥。在此示例中，将为联合身份用户 (User1) 的时长两个小时的会话获取临时安全凭证，然后使用这些凭证向 Amazon S3 发送经身份验证的请求。

- 在此练习中，您将创建具有最低权限的 IAM 用户。通过使用此 IAM 用户的凭证，可为其他用户请求临时凭证。此示例仅列出特定存储桶中的对象。创建附加了以下策略的 IAM 用户：

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

  该策略允许 IAM 用户请求临时安全凭证和用于仅列出 AWS 资源的访问权限。有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的创建您的 IAM 用户和管理员组。
- 使用 IAM 用户安全凭证测试以下示例。该示例将使用临时安全证书向 Amazon S3 发送经身份验证的请求。该示例在为联合身份用户 (User1) 请求临时安全凭证时指定了以下策略，该策略对列出特定存储桶 (`YourBucketName`) 中的对象设置了访问权限限制。您必须更新策略并提供您自己的现有存储桶名称。

```
{
  "Statement":[
    {
      "Sid":"1",
      "Action":["s3:ListBucket"],
      "Effect":"Allow",
      "Resource":"arn:aws:s3:::YourBucketName"
    }
  ]
}
```

- Example

  更新以下示例并提供在前面的联合身份用户访问策略中指定的存储桶名称。有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempFederatedCredentialsTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            ListObjectsAsync().Wait();
        }
```

```
        private static async Task ListObjectsAsync()
        {
            try
            {
                Console.WriteLine("Listing objects stored in a bucket");
                // Credentials use the default AWS SDK for .NET credential search
chain.
                // On local development machines, this is your default profile.
                SessionAWSCredentials tempCredentials =
                    await GetTemporaryFederatedCredentialsAsync();

                // Create a client by providing temporary security credentials.
                using (client = new AmazonS3Client(bucketRegion))
                {
                    ListObjectsRequest listObjectRequest = new
ListObjectsRequest();
                    listObjectRequest.BucketName = bucketName;

                    ListObjectsResponse response = await
client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);

                    Console.WriteLine("Press any key to continue...");
                    Console.ReadKey();
                }
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
            }
        }

        private static async Task<SessionAWSCredentials>
GetTemporaryFederatedCredentialsAsync()
        {
            AmazonSecurityTokenServiceConfig config = new
AmazonSecurityTokenServiceConfig();
```

```
            AmazonSecurityTokenServiceClient stsClient =
                new AmazonSecurityTokenServiceClient(
                                              config);

            GetFederationTokenRequest federationTokenRequest =
                                       new GetFederationTokenRequest();
            federationTokenRequest.DurationSeconds = 7200;
            federationTokenRequest.Name = "User1";
            federationTokenRequest.Policy = @"{
               ""Statement"":
               [
                 {
                   ""Sid"":""Stmt1311212314284"",
                   ""Action"":[""s3:ListBucket""],
                   ""Effect"":""Allow"",
                   ""Resource"":""arn:aws:s3:::" + bucketName + @"""
                  }
               ]
             }
            ";

            GetFederationTokenResponse federationTokenResponse =
                        await
 stsClient.GetFederationTokenAsync(federationTokenRequest);
            Credentials credentials = federationTokenResponse.Credentials;

            SessionAWSCredentials sessionCredentials =
                new SessionAWSCredentials(credentials.AccessKeyId,
                                          credentials.SecretAccessKey,
                                          credentials.SessionToken);
            return sessionCredentials;
        }
    }
}
```

PHP

本主题介绍如何使用 AWS SDK for PHP 的版本 3 中的类请求适用于联合身份用户和应用程序的临时安全凭证，并使用这些凭证访问 Amazon S3 中存储的资源。有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

您可以为联合身份用户和应用程序提供临时安全凭证，以便它们可以发送经身份验证的请求，从而访问您的 AWS 资源。请求这些临时凭证时，必须提供用户名和 IAM 策略（描述要授予的资源权限）。当会话持续时间过期时，这些凭证将过期。默认情况下，会话的持续时间为一个小时。可以在请求适用于联合身份用户和应用程序的临时安全凭证时为持续时间显式设置不同的值。有关临时安全凭证的更多信息，请参阅《IAM 用户指南》中的临时安全凭证。有关向联合身份用户和应用程序提供临时安全凭证的信息，请参阅提出请求。

为了提高请求适用于联合身份用户和应用程序的临时安全凭证时的安全性，建议使用仅具有所需访问权限的专用 IAM 用户。您创建的临时用户可获取的许可绝对不能超过请求临时安全证书的 IAM 用户的许可。有关联合身份的更多信息，亲参阅 AWS Identity and Access Management 常见问题。

有关适用于 Ruby 的 AWS 开发工具包 API 的更多信息，请转到适用于 Ruby 的 AWS 开发工具包 – 版本 2。

Example

以下 PHP 示例列出了指定存储桶中的密钥。在此示例中，您将获取联合身份用户 (User1) 的时长一小时的会话的临时安全凭证。然后，您使用临时安全凭证向 Amazon S3 发送经身份验证的请求。

为了提高请求适用于其他人的临时凭证时的安全性，请使用具有请求临时安全凭证的权限的 IAM 用户的安全凭证。要确保 IAM 用户仅向联合身份用户授予特定于应用程序的最低权限，还可以限制此 IAM 用户的访问权限。此示例仅列出特定存储桶中的对象。创建附加了以下策略的 IAM 用户：

```
{
  "Statement":[{
      "Action":["s3:ListBucket",
        "sts:GetFederationToken*"
      ],
      "Effect":"Allow",
      "Resource":"*"
    }
  ]
}
```

该策略允许 IAM 用户请求临时安全凭证和用于仅列出 AWS 资源的访问权限。有关如何创建 IAM 用户的更多信息，请参阅《IAM 用户指南》中的创建您的第一个 IAM 用户和管理员组。

您现在可以使用 IAM 用户安全证书测试下面的示例。该示例将使用临时安全凭证向 Amazon S3 发送经身份验证的请求。为联合身份用户 (User1) 请求临时安全凭证时，此示例指定了以下策略，该策略将访问权限限制为列出特定存储桶中的对象。请使用存储桶名称更新该策略。

```
{
  "Statement":[
    {
      "Sid":"1",
      "Action":["s3:ListBucket"],
      "Effect":"Allow",
      "Resource":"arn:aws:s3:::YourBucketName"
    }
  ]
}
```

在以下示例中，当指定策略资源时，请将 YourBucketName 替换为您的存储桶的名称：

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

// In real applications, the following code is part of your trusted code. It has
// the security credentials that you use to obtain temporary security credentials.
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Fetch the federated credentials.
$sessionToken = $sts->getFederationToken([
    'Name'              => 'User1',
    'DurationSeconds'    => '3600',
    'Policy'            => json_encode([
        'Statement' => [
            'Sid'               => 'randomstatementid' . time(),
            'Action'            => ['s3:ListBucket'],
            'Effect'            => 'Allow',
            'Resource'          => 'arn:aws:s3:::' . $bucket
        ]
    ])
]);

// The following will be part of your less trusted code. You provide temporary
```

```
// security credentials so the code can send authenticated requests to Amazon S3.

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

您可以为联合身份用户和应用程序提供临时安全凭证，以便它们可以发送经身份验证的请求，从而访问您的 AWS 资源。从 IAM 服务请求临时凭证时，您必须提供用户名和 IAM 策略（描述您需要授予的资源权限）。默认情况下，会话的持续时间为一个小时。但是，如果您使用 IAM 用户证书来请求临时证书，您可以在为联合用户和应用程序请求临时安全证书时，显式地设置其他持续时间。有关适用于联合身份用户和应用程序的临时安全凭证的信息，请参阅提出请求。

> **Note**
>
> 为了提高请求适用于联合身份用户和应用程序的临时安全凭证时的安全性，您可能需要使用仅具有所需访问权限的专用 IAM 用户。您创建的临时用户可获取的许可绝对不能超过请求临时安全证书的 IAM 用户的许可。有关更多信息，请参阅 AWS Identity and Access Management 常见问题。

Example

以下 Ruby 代码示例允许具有一组有限权限的联合身份用户列出指定存储桶中的键。

```
# Prerequisites:
```

```
#  - An existing Amazon S3 bucket.

require "aws-sdk-s3"
require "aws-sdk-iam"
require "json"

# Checks to see whether a user exists in IAM; otherwise,
# creates the user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Aws::IAM::Types::User] The existing or new user.
# @example
#   iam = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam, 'my-user')
#   exit 1 unless user.user_name
#   puts "User's name: #{user.user_name}"
def get_user(iam, user_name)
  puts "Checking for a user with the name '#{user_name}'..."
  response = iam.get_user(user_name: user_name)
  puts "A user with the name '#{user_name}' already exists."
  return response.user
# If the user doesn't exist, create them.
rescue Aws::IAM::Errors::NoSuchEntity
  puts "A user with the name '#{user_name}' doesn't exist. Creating this user..."
  response = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  puts "Created user with the name '#{user_name}'."
  return response.user
rescue StandardError => e
  puts "Error while accessing or creating the user named '#{user_name}':
 #{e.message}"
end

# Gets temporary AWS credentials for an IAM user with the specified permissions.
#
# @param sts [Aws::STS::Client] An initialized AWS STS client.
# @param duration_seconds [Integer] The number of seconds for valid credentials.
# @param user_name [String] The user's name.
# @param policy [Hash] The access policy.
# @return [Aws::STS::Types::Credentials] AWS credentials for API authentication.
# @example
#   sts = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_temporary_credentials(sts, duration_seconds, user_name,
```

```
#     {
#       'Version' => '2012-10-17',
#       'Statement' => [
#         'Sid' => 'Stmt1',
#         'Effect' => 'Allow',
#         'Action' => 's3:ListBucket',
#         'Resource' => 'arn:aws:s3:::doc-example-bucket'
#       ]
#     }
#   )
#   exit 1 unless credentials.access_key_id
#   puts "Access key ID: #{credentials.access_key_id}"
def get_temporary_credentials(sts, duration_seconds, user_name, policy)
  response = sts.get_federation_token(
    duration_seconds: duration_seconds,
    name: user_name,
    policy: policy.to_json
  )
  return response.credentials
rescue StandardError => e
  puts "Error while getting federation token: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'doc-example-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts "Name => ETag"
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
```

```
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  return true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

# Example usage:
def run_me
  region = "us-west-2"
  user_name = "my-user"
  bucket_name = "doc-example-bucket"

  iam = Aws::IAM::Client.new(region: region)
  user = get_user(iam, user_name)

  exit 1 unless user.user_name

  puts "User's name: #{user.user_name}"
  sts = Aws::STS::Client.new(region: region)
  credentials = get_temporary_credentials(sts, 3600, user_name,
    {
      "Version" => "2012-10-17",
      "Statement" => [
        "Sid" => "Stmt1",
        "Effect" => "Allow",
        "Action" => "s3:ListBucket",
        "Resource" => "arn:aws:s3:::#{bucket_name}"
      ]
    }
  )

  exit 1 unless credentials.access_key_id

  puts "Access key ID: #{credentials.access_key_id}"
  s3_client = Aws::S3::Client.new(region: region, credentials: credentials)

  exit 1 unless list_objects_in_bucket?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

相关资源

- 使用 AWS SDK 通过 Amazon S3 进行开发
- 面向 Amazon S3 Aws\S3\S3Client 类的 AWS SDK for PHP
- AWS SDK for PHP 文档

使用 REST API 提出请求

本节介绍如何使用 REST API 向 Amazon S3 终端节点发出请求。有关 Amazon S3 端点的列表，请参阅《AWS 一般参考》中的区域和端点。

为 REST API 请求构建 S3 主机名

Amazon S3 终端节点遵循如下所示的结构：

```
s3.Region.amazonaws.com
```

Amazon S3 访问点终端节点和双堆栈终端节点也遵循标准结构：

- Amazon S3 访问点 -s3-accesspoint.*Region*.amazonaws.com
- 双堆栈 – s3.dualstack.*Region*.amazonaws.com

有关 Amazon S3 区域和端点的完整列表，请参阅《Amazon Web Services 一般参考》中的 Amazon S3 端点和限额。

虚拟托管类型和路径类型请求

在使用 REST API 发出请求时，您可以使用 Amazon S3 终端节点的虚拟托管类型或路径类型 URI。有关更多信息，请参阅 存储桶的虚拟托管。

Example 虚拟托管类型请求

下面是要求从美国西部（俄勒冈）区域中名为 puppy.jpg 的存储桶删除 examplebucket 文件的虚拟托管类型请求的示例。有关虚拟托管类型请求的更多信息，请参阅虚拟托管类型请求。

```
DELETE /puppy.jpg HTTP/1.1
Host: examplebucket.s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
```

```
Authorization: authorization string
```

Example 路径类型请求

下面是同一请求的路径类型版本示例。

```
DELETE /examplebucket/puppy.jpg HTTP/1.1
Host: s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

目前，Amazon S3 在所有 AWS 区域中同时支持虚拟托管类型和路径类型 URL 访问。但是，路径类型 URL 将来会停用。有关更多信息，请参阅下面的重要提示。

有关路径类型请求的更多信息，请参阅路径类型请求。

> **Important**
>
> 更新（2020 年 9 月 23 日）– 为确保客户有时间转换到虚拟托管类型 URL，我们决定推迟弃用路径类型 URL。有关更多信息，请参阅 AWS 新闻博客中的 Amazon S3 路径弃用计划 – 案例的其余部分。

通过使用 REST API 向双堆栈终端节点发出请求

在使用 REST API 时，您可以通过使用虚拟托管类型或路径类型终端节点名称 (URI) 直接访问双堆栈终端节点。所有 Amazon S3 双堆栈终端节点名称中都包含区域。与标准的仅支持 IPv4 的终端节点不同，虚拟托管类型和路径类型终端节点都使用特定于区域的终端节点名称。

Example 虚拟托管类型双堆栈终端节点请求

如下例所示，您可以在 REST 请求中使用虚拟托管类型终端节点，该示例从美国西部（俄勒冈）区域中名为 `puppy.jpg` 的存储桶检索 `examplebucket` 对象。

```
GET /puppy.jpg HTTP/1.1
Host: examplebucket.s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example 路径类型双堆栈终端节点请求

如下例所示，您也可以在请求中使用路径类型终端节点。

```
GET /examplebucket/puppy.jpg HTTP/1.1
Host: s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

有关双堆栈终端节点的更多信息，请参阅使用 Amazon S3 双堆栈端点。

有关如何使用 REST API 发出请求的更多信息，请参阅以下主题。

主题

- 存储桶的虚拟托管
- 请求重定向和 REST API

存储桶的虚拟托管

虚拟托管是指从单个 Web 服务器为多个网站提供服务的做法。区分 Amazon S3 REST API 请求中的站点的一种方法是通过使用请求-URI 的显式主机名，而不只是 URI 的路径名称部分。普通的 Amazon S3 REST 请求使用“请求-URI”路径的第一个斜杠分隔部分指定存储桶。相反，您可以通过使用 HTTP `Host` 标头，在 REST API 调用中使用 Amazon S3 虚拟托管对桶进行寻址。事实上，Amazon S3 会将 `Host` 解释为可在 `https://`*`bucket-name`*`.s3.`*`region-code`*`.amazonaws.com` 上自动访问大多数存储桶（对于有限类型的请求）。有关 Amazon S3 区域和端点的完整列表，请参阅《Amazon Web Services 一般参考》中的 Amazon S3 端点和限额。

虚拟托管还有其他好处。通过使用注册域名来命名您的存储桶，并将该名称设为 Amazon S3 的 DNS 别名，您可以完全自定义 Amazon S3 资源的 URL，例如：`http://my.`*`bucket-name`*`.com/`。您还可以发布到存储桶的虚拟服务器的“根目录”。此功能很重要，因为许多现有应用程序在此标准位置搜索文件。例如，`favicon.ico`、`robots.txt` 和 `crossdomain.xml` 都应可在根目录下找到。

> **Important**
>
> 当通过 SSL 使用虚拟托管类型桶时，SSL 通配符证书仅匹配不包含句点 (`.`) 的桶。要解决此问题，请使用 HTTP 或编写自己的证书验证逻辑。有关更多信息，请参阅 AWS 新闻博客 上的 Amazon S3 路径弃用计划。

主题

路径类型请求

目前，Amazon S3 在所有 AWS 区域中同时支持虚拟托管类型和路径类型 URL 访问。但是，路径类型 URL 将来会停用。有关更多信息，请参阅下面的重要提示。

在 Amazon S3 中，路径类型 URL 使用以下格式：

```
https://s3.region-code.amazonaws.com/bucket-name/key-name
```

例如，如果您在美国西部（俄勒冈）区域中创建一个名为 `amzn-s3-demo-bucket1` 的存储桶，并希望访问该存储桶中的 `puppy.jpg` 对象，则可使用以下路径类型 URL：

```
https://s3.us-west-2.amazonaws.com/amzn-s3-demo-bucket1/puppy.jpg
```

> **Important**
>
> 更新（2020 年 9 月 23 日）– 为确保客户有时间转换到虚拟托管类型 URL，我们决定推迟弃用路径类型 URL。有关更多信息，请参阅 AWS 新闻博客中的 Amazon S3 路径弃用计划 – 案例的其余部分。

> **Warning**
>
> 托管将通过 Web 浏览器访问的网站内容时，请避免使用路径样式 URL，因为这可能会干扰浏览器的同源安全模型。要托管网站内容，我们建议您使用 S3 网站端点或 CloudFront 分配。有

关更多信息，请参阅网站端点和《AWS 透视指导模式》中的将基于 React 的单页应用程序部署到 Amazon S3 和 CloudFront。

虚拟托管类型请求

在虚拟托管类型 URI 中，存储桶名称是 URL 中域名的一部分。

Amazon S3 虚拟托管类型 URL 使用以下格式：

```
https://bucket-name.s3.region-code.amazonaws.com/key-name
```

在此示例中，amzn-s3-demo-bucket1 为存储桶名称，美国西部（俄勒冈）为区域，puppy.png 为密钥名称：

```
https://amzn-s3-demo-bucket1.s3.us-west-2.amazonaws.com/puppy.png
```

HTTP **Host** 标头桶规范

只要您的 GET 请求不使用 SSL 终端节点，您就可以通过使用 HTTP Host 标头为请求指定存储桶。REST 请求中的 Host 标头解释如下：

- 如果 Host 标头被省略，或其值为 s3.*region-code*.amazonaws.com，则该请求的存储桶将是请求-URI 的第一个斜杠分隔的部分，而且该请求的键将是请求-URI 的其余部分。这是常用方法，如本部分第一个和第二个示例所示。省略 Host 标头仅对于 HTTP 1.0 请求有效。
- 否则，如果 Host 标头的值以 .s3.*region-code*.amazonaws.com 结尾，则存储桶名称是 Host 标头值中从开头到 .s3.*region-code*.amazonaws.com 的部分。该请求的键值为“Request-URI”。此解释将存储桶公开为 .s3.*region-code*.amazonaws.com 的子域，如本部分第三和第四个示例所示。
- 否则，请求的存储桶是 Host 标头的小写值，请求的键值是“Request-URI”。如果注册的 DNS 名称与桶名称相同，并且已将该名称配置为 Amazon S3 的规范名称 (CNAME) 别名，则此解释很有用。注册域名和配置 CNAME DNS 记录的过程不在本指南的范围内，但本节中的最后一个示例展示了结果。

示例

本节提供示例 URL 和请求。

Example – 路径类型 URL 和请求

此示例使用以下内容：

- 存储桶名称 – example.com
- 区域 – 美国东部（弗吉尼亚北部）
- 键名 – homepage.html

URL 如下：

```
http://s3.us-east-1.amazonaws.com/example.com/homepage.html
```

请求如下：

```
GET /example.com/homepage.html HTTP/1.1
Host: s3.us-east-1.amazonaws.com
```

具有 HTTP 1.0 且省略 Host 标头的请求如下：

```
GET /example.com/homepage.html HTTP/1.0
```

有关 DNS 兼容名称的信息，请参阅限制。有关密钥的更多信息，请参阅密钥。

Example – 虚拟托管类型 URL 和请求

此示例使用以下内容：

- 桶名称 - amzn-s3-demo-bucket1
- 区域 – 欧洲（爱尔兰）
- 键名 - homepage.html

URL 如下：

```
http://amzn-s3-demo-bucket1.s3.eu-west-1.amazonaws.com/homepage.html
```

请求如下：

```
GET /homepage.html HTTP/1.1
```

```
Host: amzn-s3-demo-bucket1.s3.eu-west-1.amazonaws.com
```

Example – CNAME 别名方法

要使用此方法，您必须将 DNS 名称配置为 *bucket-name*.s3.us-east-1.amazonaws.com 的 CNAME 别名。有关更多信息，请参阅 使用 CNAME 记录自定义 Amazon S3 URL。

此示例使用以下内容：

- 存储桶名称 – example.com
- 键名称 - homepage.html

URL 如下：

```
http://www.example.com/homepage.html
```

示例如下：

```
GET /homepage.html HTTP/1.1
Host: www.example.com
```

使用 CNAME 记录自定义 Amazon S3 URL

根据您的需要，您可能不希望 s3.*region-code*.amazonaws.com 显示在网站或服务中。例如，如果您在 Amazon S3 上托管网站映像，您可能会首选 http://images.example.com/，而不是 http://images.example.com.s3.us-east-1.amazonaws.com/。具有 DNS 兼容名称的任何存储桶都可按以下方式引用： http://*BucketName*.s3.*Region*.amazonaws.com/[*Filename*]，例如 http://images.example.com.s3.us-east-1.amazonaws.com/mydog.jpg。通过使用 CNAME，您可以将 images.example.com 映射到 Amazon S3 主机名，使上面的 URL 变成 http://images.example.com/mydog.jpg。

存储桶名称必须与 CNAME 相同。例如，如果您创建一个 CNAME 以将 images.example.com 映射到 images.example.com.s3.us-east-1.amazonaws.com，则 http://images.example.com/filename 和 http://images.example.com.s3.us-east-1.amazonaws.com/filename 两者将相同。

CNAME DNS 记录应将您的域名别名设置为适当的虚拟托管类型主机名。例如，如果存储桶名称和域名是 images.example.com 而存储桶位于美国东部（弗吉尼亚北部）区域中，则 CNAME 记录的别名应为 images.example.com.s3.us-east-1.amazonaws.com。

```
images.example.com CNAME    images.example.com.s3.us-east-1.amazonaws.com.
```

Amazon S3 使用主机名来确定存储桶名称。所以 CNAME 和存储桶名称必须相同。例如，假定您配置了 www.example.com 作为 www.example.com.s3.us-east-1.amazonaws.com 的别名记录。当您访问http://www.example.com时，Amazon S3 会收到如以下所示的请求：

Example

```
GET / HTTP/1.1
Host: www.example.com
Date: date
Authorization: signatureValue
```

Amazon S3 仅查看原始主机名 www.example.com 且不知道用于解决请求的 CNAME 映射。

您可以在 CNAME 别名中使用任何 Amazon S3 端点。例如，s3.ap-southeast-1.amazonaws.com 可用于 CNAME 别名中。有关终端节点的更多信息，请参阅请求终端节点。要使用自定义域创建静态网站，请参阅教程：使用注册到 Route 53 的自定义域配置静态网站。

> **Important**
>
> 将自定义 URL 与 CNAME 结合使用时，您需要确保对于您配置的任何 CNAME 或别名记录都存在匹配的桶。例如，如果您为 www.example.com 和 login.example.com 创建 DNS 条目以使用 S3 发布 Web 内容，将需要同时创建两个桶 www.example.com 和 login.example.com。
>
> 当 CNAME 或别名记录配置为指向没有匹配桶的 S3 端点时，任何 AWS 用户都可以创建该桶并在配置的别名下发布内容，即使拥有权不同也是如此。
>
> 出于同样的原因，我们建议您在删除桶时更改或移除相应的 CNAME 或别名。

如何将主机名与 Amazon S3 桶关联

使用 CNAME 将主机名与 Amazon S3 桶关联

1. 选择属于您控制的域的主机名。

   本示例使用 images 域的 example.com 子域。
2. 创建与主机名匹配的存储桶。

在本示例中，主机名和存储桶名称是 images.example.com。存储桶名称必须与主机名精确 匹配。

3. 创建一条 CNAME DNS 记录，它将主机名定义为 Amazon S3 桶的别名。

   例如：

   ```
   images.example.com CNAME images.example.com.s3.us-west-2.amazonaws.com
   ```

   > **Important**
   >
   > 出于请求路由原因，CNAME DNS 记录必须严格按照上述示例所示进行定义。否则，它可能显示为正常运行，但将最终导致不可预测的行为。

   配置 CNAME DNS 记录的过程取决于您的 DNS 服务器或 DNS 提供商。有关特定信息，请参阅您的服务器文档或与您的供应商联系。

限制

HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 特征。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

向后兼容性

以下各节介绍了 Amazon S3 向后兼容性的各个方面，这些方面与路径类型和虚拟托管类型 URL 请求相关。

传统终端节点

某些区域支持传统终端节点。您可能会在服务器访问日志或 AWS CloudTrail 日志中看到这些端点。有关更多信息，请查看以下信息。有关 Amazon S3 区域和端点的完整列表，请参阅《Amazon Web Services 一般参考》中的 Amazon S3 端点和限额。

> **Important**
>
> 尽管您可能会在日志中看到传统终端节点，但我们建议您始终使用标准终端节点语法访问存储桶。
>
> Amazon S3 虚拟托管类型 URL 使用以下格式：

```
https://bucket-name.s3.region-code.amazonaws.com/key-name
```

在 Amazon S3 中，路径类型 URL 使用以下格式：

```
https://s3.region-code.amazonaws.com/bucket-name/key-name
```

s3-区域

一些较旧的 Amazon S3 区域支持在 s3 和区域代码之间包含短划线 (-) 而不是点（例如 s3.us-west-2）的端点（例如 s3-us-west-2）。如果您的存储桶位于这些区域之一，您可能会在服务器访问日志或 CloudTrail 日志中看到以下终端节点格式：

```
https://bucket-name.s3-region-code.amazonaws.com
```

在此示例中，存储桶名称为 amzn-s3-demo-bucket1，区域为美国西部（俄勒冈）：

```
https://amzn-s3-demo-bucket1.s3-us-west-2.amazonaws.com
```

传统全局端点

对于某些区域，可以使用传统全局端点来构建未指定特定于区域的端点的请求。传统全局终端节点如下所示：

```
bucket-name.s3.amazonaws.com
```

在服务器访问日志或 CloudTrail 日志中，您可能会看到使用传统全局终端节点的请求。在此示例中，桶名称为 amzn-s3-demo-bucket1，传统全局端点为：

```
https://amzn-s3-demo-bucket1.s3.amazonaws.com
```

美国东部（弗吉尼亚州北部）的虚拟托管类型请求

原定设置情况下，使用传统全局端点发出的请求会转到美国东部（弗吉尼亚州北部）。因此，传统全局终端节点有时用于替代美国东部（弗吉尼亚北部）的区域终端节点。如果您在美国东部（弗吉尼亚州北部）创建桶并使用全局端点，则原定设置情况下，Amazon S3 会将您的请求路由到此区域。

其他区域的虚拟托管类型请求

传统全局端点也用于其他受支持的区域中的虚拟托管类型请求。当您在 2019 年 3 月 20 日之前发布的区域中创建桶并使用传统全局端点时，Amazon S3 将更新 DNS 记录以将请求重新路由到正确的位置，这可能会花费一些时间。同时，将应用原定设置规则，您的虚拟托管类型请求将转到美国东部（弗吉尼亚州北部）区域。然后，Amazon S3 将使用 HTTP 307 临时重定向将它重定向到正确的区域。

对于在 2019 年 3 月 20 日之后发布的区域中的 S3 桶，DNS 服务器不会将您的请求直接路由到您的桶所在的 AWS 区域。它而是会返回 HTTP 400 错误请求错误。有关更多信息，请参阅 提出请求。

路径类型请求

对于美国东部（弗吉尼亚州北部）区域，可以将传统全局端点用于路径类型请求。

对于所有其他区域，路径类型语法要求您在尝试访问存储桶时使用特定于区域的终端节点。如果您尝试访问具有传统全局端点的桶或与桶所在区域的端点不同的其他端点，您会收到 HTTP 响应代码“307 临时重定向”错误和一条消息（指示您资源的正确 URI）。例如，如果您将 `https://s3.amazonaws.com/bucket-name` 用于在美国西部（俄勒冈）区域创建的存储桶，您将收到“HTTP 307 Temporary Redirect (HTTP 307 临时重定向)”错误。

请求重定向和 REST API

主题

- 重定向和 HTTP 用户代理
- 重定向和 100-继续
- 重定向示例

此部分介绍如何使用 Amazon S3 REST API 处理 HTTP 重定向。有关 Amazon S3 重定向的一般信息，请参阅《Amazon Simple Storage Service API 参考》中的提出请求。

重定向和 HTTP 用户代理

使用 Amazon S3 REST API 的程序既可以处理应用程序层上的重定向，也可以处理 HTTP 层上的重定向。许多 HTTP 客户端库和用户代理经过配置可以自动正确处理重定向；然而，许多其他 HTTP 客户端库和用户代理的重定向实施会不正确或不完整。

在依赖库完成重定向要求之前，请测试以下案例：

- 验证所有 HTTP 请求标头都已正确地包含在重定向的请求中 (收到重定向后的第二个请求)，包括诸如授权和日期等 HTTP 标准。
- 验证诸如 PUT 和 DELETE 等非 GET 的重定向运行正常。

- 验证大型 PUT 请求正确地遵循了重定向。
- 如果需要很长时间才能收到 100-继续响应，请验证 PUT 请求正确地遵循了重定向。

当 HTTP 请求方法不是 GET 或 HEAD 时，严格遵循 RFC 2616 的 HTTP 用户代理可能会在遵循重定向之前，请求显式确认。自动遵循 Amazon S3 生成的重定向通常是安全的，因为系统只会发出指向 amazonaws.com 域中主机的重定向，并且重定向请求的效果与原始请求的效果相同。

重定向和 100-继续

要简化重定向处理过程、提高工作效率以及避免重复支付与发送重定向请求正文相关的费用，请将您的应用程序配置为对 PUT 操作使用 100-继续。当应用程序使用 100-继续时，它不会发送请求正文，直到收到认可。如果根据标头拒绝了消息，则不会发送消息的正文。有关 100-继续的更多信息，请参阅 RFC 2616 Section 8.2.3

> **Note**
>
> 根据 RFC 2616，在未知的 HTTP 服务器中使用 `Expect: Continue` 时，您应该直接发送请求正文，不要等待某个无限期的时段。这是因为某些 HTTP 服务器不能识别 100-继续。但是，Amazon S3 能够识别您的请求是否包含了 `Expect: Continue`，并且将使用临时的 100-继续状态或最终的状态代码进行响应。此外，收到临时的 100 继续操作指示后，不会发生重定向错误。这可以帮助您避免在编写请求正文时，收到重定向响应。

重定向示例

本节提供了使用 HTTP 重定向和 100-继续执行客户端和服务器交互的示例。

下面是用于 `quotes.s3.amazonaws.com` 存储桶的示例 PUT。

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 将返回以下内容：

```
HTTP/1.1 307 Temporary Redirect
```

```
Location: http://quotes.s3-4c25d83b.amazonaws.com/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT

Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the
  specified temporary endpoint. Continue to use the
  original request endpoint for future requests.
  </Message>
  <Endpoint>quotes.s3-4c25d83b.amazonaws.com</Endpoint>
  <Bucket>quotes</Bucket>
</Error>
```

客户端将遵循重定向响应并向 quotes.s3-4c25d83b.amazonaws.com 临时终端节点发布新的请求。

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 将返回 100-继续，以指示客户端应继续发送请求正文。

```
HTTP/1.1 100 Continue
```

客户端将发送请求正文。

```
ha ha\n
```

Amazon S3 将返回最终响应。

```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2007 22:18:48 GMT
```

```
ETag: "a2c8d6b872054293afd41061e93bc289"
Content-Length: 0
Server: AmazonS3
```

使用 AWS CLI 进行 Amazon S3 开发

按照以下步骤下载和配置 AWS Command Line Interface (AWS CLI)。

有关 Amazon S3 AWS CLI 命令的列表，请参阅《AWS CLI 命令参考》中的以下页面：

- S3
- s3api
- s3control

> **Note**
>
> AWS 中的服务（如 Amazon S3）要求您在访问时提供凭证。然后该服务才能确定您是否有权访问该服务所拥有的资源。控制台要求您的密码。您可以为您的 AWS 账户 创建访问密钥以访问 AWS CLI 或 API。但是，我们不建议使用您的 AWS 账户 的凭证访问 AWS。相反，我们建议您使用 AWS Identity and Access Management (IAM)。创建 IAM 用户，将该用户添加到具有管理权限的 IAM 组，然后向您创建的 IAM 用户授予管理权限。然后，您就可以使用专门的 URL 和该 IAM 用户的凭证来访问 AWS。有关说明，请转到《IAM 用户指南》中的创建您的第一个 IAM 用户和管理员组。

设置 AWS CLI

1. 下载并配置 AWS CLI。有关说明，请参阅《AWS Command Line Interface 用户指南》中的以下主题：

   - 开始设置 AWS Command Line Interface
   - 配置 AWS Command Line Interface

2. 在 AWS CLI 配置文件中为管理员用户添加一个命名配置文件。在执行 AWS CLI 命令时，您将使用此配置文件。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的 AWS CLI 的命名配置文件。

   ```
   [adminuser]
   ```

```
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

有关可用 AWS 区域的列表，请参阅《AWS 一般参考》中的区域和端点。

3. 在命令提示符处输入以下命令来验证设置。

   - 尝试 `help` 命令来验证您的计算机上是否安装了 AWS CLI：

     ```
     aws help
     ```

   - 使用您刚创建的 `adminuser` 凭证运行 S3 命令。为此，请将 `--profile` 参数添加到您的命令，以指定配置文件名称。在本示例中，`ls` 命令将列出您的账户中的存储桶。AWS CLI 使用 `adminuser` 凭证来验证请求。

     ```
     aws s3 ls --profile adminuser
     ```

使用 AWS SDK 通过 Amazon S3 进行开发

AWS 软件开发工具包 (SDK) 适用于许多常用编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

> **Note**
>
> 您可以使用 AWS Amplify 进行 Web 和移动应用程序的端到端全栈开发。Amplify Storage 将文件存储和管理功能无缝集成到基于 Amazon S3 构建的前端 Web 和移动应用程序中。有关更多信息，请参阅《Amplify 用户指南》中的 Storage。

将此服务与 AWS SDK 结合使用

AWS 软件开发工具包 (SDK) 适用于许多常用编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档	代码示例
AWS SDK for C++	AWS SDK for C++ 代码示例

SDK 文档	代码示例
AWS CLI	AWS CLI 代码示例
AWS SDK for Go	AWS SDK for Go 代码示例
AWS SDK for Java	AWS SDK for Java 代码示例
AWS SDK for JavaScript	AWS SDK for JavaScript 代码示例
AWS SDK for Kotlin	AWS SDK for Kotlin 代码示例
AWS SDK for .NET	AWS SDK for .NET 代码示例
AWS SDK for PHP	AWS SDK for PHP 代码示例
AWS Tools for PowerShell	Tools for PowerShell 代码示例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 代码示例
AWS SDK for Ruby	AWS SDK for Ruby 代码示例
AWS SDK for Rust	AWS SDK for Rust 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
AWS SDK for Swift	AWS SDK for Swift 代码示例

有关特定于此服务的示例，请参阅适用于使用 AWS 软件开发工具包的 Amazon S3 的代码示例。

> **示例可用性**
>
> 找不到所需的内容？ 通过使用此页面底部的提供反馈链接请求代码示例。

SDK 编程接口

每个 AWS SDK 都提供一个或多个编程接口来与 Amazon S3 结合使用。每个 SDK 都为 Amazon S3 提供一个低级别接口，其方法与 API 操作非常相似。一些 SDK 为 Amazon S3 提供高级别接口，这类抽象旨在简化常见的应用场景。

例如，当您使用低级别 API 操作来执行分段上传时，您需要使用一个操作来启动上传，使用另一个操作来上传各个分段，然后使用最后一个操作来完成上传。高级别分段上传 API 操作可让您在单个 API 调用中执行上传所需的所有操作。有关示例，请参阅使用分段上传操作上传对象。

低级别 API 操作可以更好地控制上传。如果您需要暂停和恢复上传，在上传期间更改分段的大小，或者在事先不知道数据大小的情况下开始上传，建议您使用低级别 API。

在请求身份验证中指定签名版本

Amazon S3 在大多数 AWS 区域 中只支持 AWS Signature Version 4。不过，在某些较旧的 AWS 区域 中，Amazon S3 同时支持 Signature Version 4 和 Signature Version 2。但是，签名版本 2 将要被关闭（弃用）。有关结束对签名版本 2 支持的更多信息，请参阅AWS已为 Amazon S3 关闭（弃用）Signature Version 2。

有关所有 Amazon S3 区域的列表以及这些区域支持的 Signature Version，请参阅《AWS 一般参考》中的区域和终端节点。

对于所有 AWS 区域，默认情况下，AWS 开发工具包使用 Signature Version 4 对请求进行身份验证。如果使用 2016 年 5 月之前发布的 AWS 开发工具包，您可能需要请求 Signature Version 4，如下表所示。

开发工具包	请求使用签名版本 4 进行请求身份验证
AWS CLI	对于默认配置文件，运行以下命令： `$ aws configure set default.s3.signature_version s3v4` 对于自定义配置文件，运行以下命令： `$ aws configure set profile.your_profile_name.s3.signature_version s3v4`
Java 软件开发工具包	在代码中添加以下内容： `System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");` 或者在命令行中指定以下内容：

开发工具包	请求使用签名版本 4 进行请求身份验证
	`-Dcom.amazonaws.services.s3.enableV4`
JavaScript 开发工具包	在构建客户端时，将 `signatureVersion` 参数设置为 `v4`： `var s3 = new AWS.S3({signatureVersion: 'v4'});`
PHP 开发工具包	在为 PHP 开发工具包 v2 构建 Amazon S3 服务客户端时，将 `signature` 参数设置为 `v4`： `<?php` `$client = S3Client::factory([` `'region' => 'YOUR-REGION',` `'version' => 'latest',` `'signature' => 'v4'` `]);` 使用 PHP 开发工具包 v3 时，在构建 Amazon S3 服务客户端期间将 `signature_version` 参数设置为 `v4`： `<?php` `$s3 = new Aws\S3\S3Client([` `'version' => '2006-03-01',` `'region' => 'YOUR-REGION',` `'signature_version' => 'v4'` `]);`
Python-Boto 开发工具包	在默认配置文件 boto 中指定以下内容： `[s3] use-sigv4 = True`

开发工具包	请求使用签名版本 4 进行请求身份验证
Ruby 开发工具包	Ruby 开发工具包 – 版本 1：在构建客户端时，将 `:s3_signature_version` 参数设置为 `:v4`： `s3 = AWS::S3::Client.new(:s3_signature_version => :v4)` Ruby 开发工具包 – 版本 3：在构建客户端时，将 `signature_version` 参数设置为 `v4`： `s3 = Aws::S3::Client.new(signature_version: 'v4')`
.NET 开发工具包	在创建 Amazon S3 客户端之前将以下内容添加到代码中： `AWSConfigsS3.UseSignatureVersion4 = true;` 或将以下内容添加到配置文件中： `<appSettings>` `  <add key="AWS.S3.UseSignatureVersion4" value="true" />` `</appSettings>`

AWS已为 Amazon S3 关闭（弃用） Signature Version 2

但是，Amazon S3 即将关闭（弃用）签名版本 2。之后，Amazon S3 仅接受使用签名版本 4 签名的 API 请求。

本部分提供有关结束签名版本 2 支持的常见问题解答。

什么是签名版本 2/4，以及这对于签名请求意味着什么？

签名版本 2 或签名版本 4 签名过程用于对 Amazon S3 API 请求进行身份验证。签名请求使得 Amazon S3 可以确定谁在发送请求并保护您的请求免遭恶意人士侵害。

有关签署 AWS 请求的更多信息，请参阅《AWS 一般参考》中的签署 AWS API 请求。

进行了哪些更新？

我们当前支持使用签名版本 2 和签名版本 4 过程签名的 Amazon S3 API 请求。在此之后，Amazon S3 将仅接受使用签名版本 4 签名的请求。

有关签署 AWS 请求的更多信息，请参阅《AWS 一般参考》中的签名版本 4 中的变化。

为什么进行这些更新？

签名版本 4 使用签名密钥而非您的秘密访问密钥，提供了更好的安全性。当前所有 AWS 区域中都支持 Signature Version 4，而只有在 2014 年 1 月之前推出的区域中支持 Signature Version 2。此更新使得我们可以在所有区域中提供更为一致的体验。

如何确保我使用的是签名版本 4，需要进行哪些更新？

对请求进行签名所用的签名版本通常由客户端上的工具或开发工具包设置。默认情况下，AWS 开发工具包的最新版本使用 Signature Version 4。对于第三方软件，请联系您软件的相应支持团队以确认所需的版本。如果您向 Amazon S3 发送直接 REST 调用，则必须修改应用程序以使用签名版本 4 签名过程。

有关在转向 Signature Version 4 时所用 AWS 开发工具包版本的信息，请参阅 从签名版本 2 转向签名版本 4。

有关将 Signature Version 4 与 Amazon S3 REST API 结合使用的信息，请参阅《Amazon Simple Storage Service API 参考》中的对请求进行身份验证 (AWS Signature Version 4)。

如果我不更新会怎样？

在此之后使用签名版本 2 签名的请求将无法通过 Amazon S3 的身份验证。请求者将看到错误，说明请求必须使用签名版本 4 签名。

如果我使用要求我签名 7 天以上的预签名 URL，是否应进行更改？

如果您使用了要求您签名 7 天以上的预签名 URL，目前无需任何操作。您可以继续使用 AWS Signature Version 2 对预签名 URL 进行签名并通过身份验证。我们将后续跟进并提供更多详情，说明如何为预签名 URL 场景迁移签名版本 4。

更多信息

- 有关使用 Signature Version 4 的更多信息，请参阅签署 AWS API 请求。
- 在签名版本 4 中的变化中可查看签名版本 2 与签名版本 4 之间的改动列表。

- 查看 AWS 论坛中的帖子：AWS Signature Version 4 将取代 AWS Signature Version 2 用于签署 Amazon S3 API 请求。
- 如果您有任何问题或疑虑，请联系 AWS Support。

从签名版本 2 转向签名版本 4

如果您为 Amazon S3 API 请求身份验证使用签名版本 2，则应改为使用签名版本 4。对签名版本 2 的支持将结束，如AWS已为 Amazon S3 关闭（弃用） Signature Version 2中所述。

有关将 Signature Version 4 与 Amazon S3 REST API 结合使用的信息，请参阅《Amazon Simple Storage Service API 参考》中的对请求进行身份验证 (AWS Signature Version 4)。

下表列出了使用签名版本 4 (SigV4) 所需的开发工具包最低版本列表。如果您使用通过 AWS Java、JavaScript (Node.js) 或 Python (Boto/CLI)开发工具包预签名 URL，则必须设置正确的 AWS 区域 并在客户端配置中设置 Signature Version 4。有关在客户端配置中设置 `SigV4` 的信息，请参阅在请求身份验证中指定签名版本。

如果您使用此开发工具包/产品	请升级至此开发工具包版本	客户端是否需要进行代码更改以使用 Sigv4？	指向开发工具包文档的链接
AWS SDK for Java v1	升级至 Java 1.11.201+ 或 v2。	是	在请求身份验证中指定签名版本
AWS SDK for Java v2	无需开发工具包升级。	否	AWS SDK for Java
AWS SDK for .NET v1	升级至 3.1.10 或更高版本。	是	AWS SDK for .NET
AWS SDK for .NET v2	升级至 3.1.10 或更高版本。	否	AWS SDK for .NET v2

如果您使用此开发工具包/产品	请升级至此开发工具包版本	客户端是否需要进行代码更改以使用 Sigv4？	指向开发工具包文档的链接
AWS SDK for .NET v3	升级至 3.3.0.0 或更高版本。	是	AWS SDK for .NET v3
AWS SDK for JavaScript v1	升级至 2.68.0 或更高版本。	是	AWS SDK for JavaScript
AWS SDK for JavaScript v2	升级至 2.68.0 或更高版本。	是	AWS SDK for JavaScript
AWS SDK for JavaScript v3	当前无需任何操作。2019 年 3 季度升级至主要版本 V3。	否	AWS SDK for JavaScript
AWS SDK for PHP v1	建议升级至最新版本的 PHP，或者至少升级至已在 S3 客户端的配置中将签名参数设置为 v4 的 v2.7.4。	是	AWS SDK for PHP

如果您使用此开发工具包/产品	请升级至此开发工具包版本	客户端是否需要进行代码更改以使用 Sigv4？	指向开发工具包文档的链接
AWS SDK for PHP v2	建议升级至最新版本的 PHP，或者至少升级至已在 S3 客户端的配置中将签名参数设置为 v4 的 v2.7.4。	否	AWS SDK for PHP
AWS SDK for PHP v3	无需开发工具包升级。	否	AWS SDK for PHP
Boto2	升级至 Boto2 v2.49.0。	是	Boto 2 升级
Boto3	升级至 1.5.71 (Botocore)、1.4.6 (Boto3)。	是	Boto 3 – 适用于 Python 的AWS开发工具包
AWS CLI	升级至 1.11.108。	是	AWS Command Line Interface
AWS CLI v2（预览）	无需开发工具包升级。	否	AWS Command Line Interface 版本 2
AWS SDK for Ruby v1	升级至 Ruby V3。	是	AWS 的 Ruby V3
AWS SDK for Ruby v2	升级至 Ruby V3。	是	AWS 的 Ruby V3

如果您使用此开发工具包/产品	请升级至此开发工具包版本	客户端是否需要进行代码更改以使用 Sigv4？	指向开发工具包文档的链接
AWS SDK for Ruby v3	无需开发工具包升级。	否	AWS 的 Ruby V3
Go	无需开发工具包升级。	否	AWS SDK for Go
C++	无需开发工具包升级。	否	AWS SDK for C++

AWS Tools for Windows PowerShell 或者 AWS Tools for PowerShell Core

如果您使用的模块版本早于 3.3.0.0，则必须升级到 3.3.0.0。

要获取版本信息，请使用 `Get-Module` cmdlet：

```
Get-Module -Name AWSPowershell
Get-Module -Name AWSPowershell.NetCore
```

要更新 3.3.0.0 版本，请使用 `Update-Module` cmdlet：

```
Update-Module -Name AWSPowershell
Update-Module -Name AWSPowershell.NetCore
```

您可以使用有效期超过 7 天的预签名 URL 以将签名版本 2 流量发送到其上。

使用 REST API 进行 Amazon S3 开发

Amazon S3 架构的设计与编程语言无关，它使用我们支持的接口来存储和检索数据元。

Amazon S3 当前提供 REST 接口。使用 REST，元数据将在 HTTP 标头中返回。由于我们仅支持最大 4 KB 的 HTTP 请求 (不包括正文)，因此您能提供的元数据量是受限的。REST API 是面向 Amazon S3 的 HTTP 接口。借助 REST，您可以使用标准的 HTTP 请求创建、提取和删除存储桶和对象。

您可以借助任何支持 HTTP 的工具包来使用 REST API。只要对象是匿名可读的，您甚至可以使用浏览器来提取它们。

REST API 使用标准的 HTTP 标头和状态代码，以使标准的浏览器和工具包按预期工作。在某些区域中，我们向 HTTP 添加了功能 (例如，我们添加了标头来支持访问控制)。在这些情况下，我们已尽最大努力使添加的新功能与标准的 HTTP 使用样式相匹配。

有关使用 REST API 发送请求的更多信息，请参阅使用 REST API 提出请求。有关使用 REST API 时应该记住的一些注意事项，请参阅以下主题。

有关 Amazon S3 REST API 的更多信息，请参阅 Amazon Simple Storage Service API 参考。

主题

请求路由选择

向使用 CreateBucket API 创建的存储桶发送请求且包含 CreateBucketConfiguration 的程序必须支持重新导向。此外，不考虑 DNS TTL 的某些客户端可能会遇到问题。

本节描述了在设计要和 Amazon S3 结合使用的服务或应用程序时，应考虑的路由选择和 DNS 问题。

请求重定向和 REST API

Amazon S3 使用域名系统 (DNS) 将请求路由到可以处理它们的设备。此系统处于有效的工作状态，但可能会发生临时路由错误。如果请求进入错误的 Amazon S3 位置，Amazon S3 将使用临时重定向进行响应，以告知请求者将请求重新发送到新的终端节点。如果请求的格式不正确，Amazon S3 将使用永久重定向来提供有关如何正确执行请求的说明。

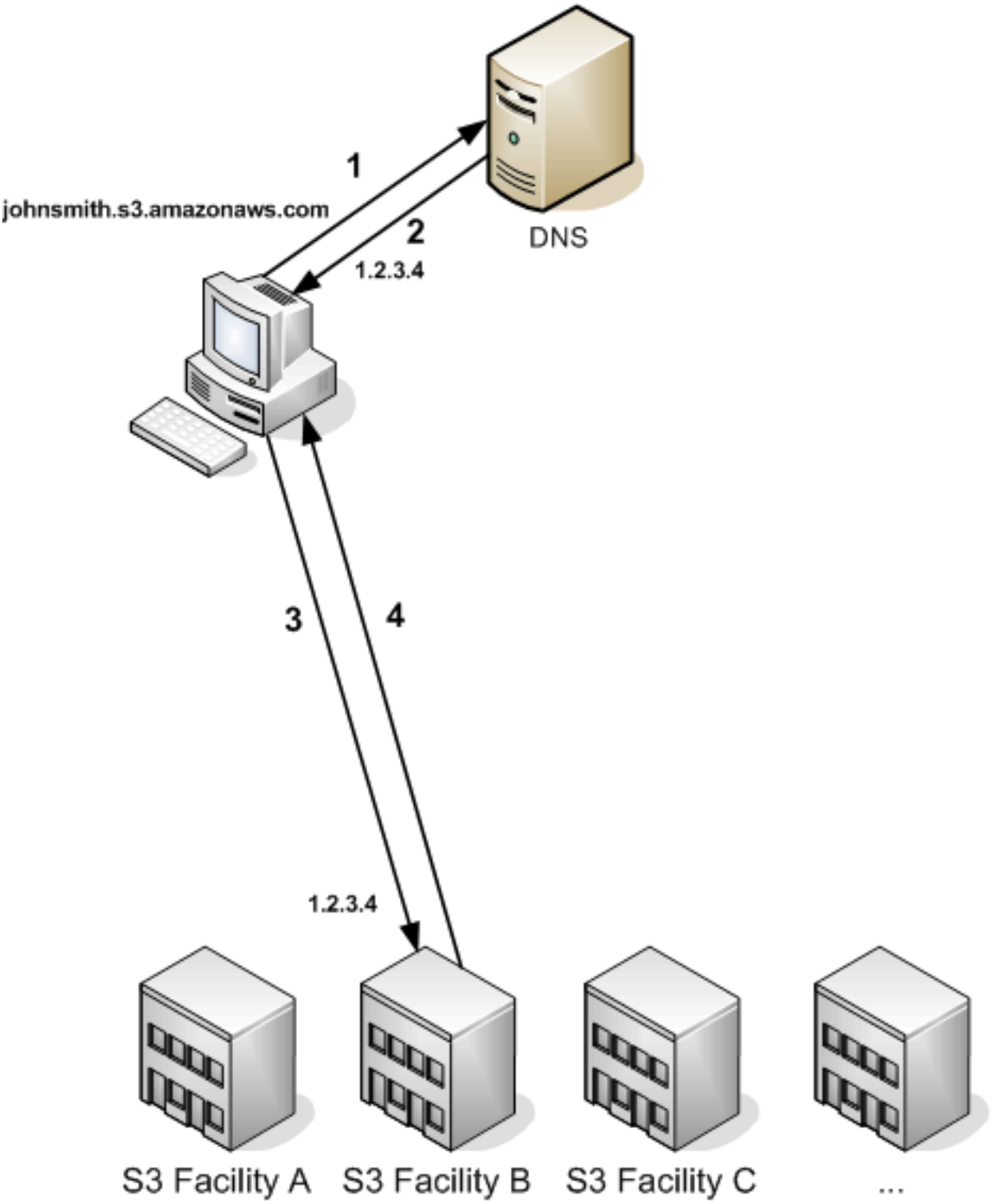
> **Important**
>
> 要使用此功能，您必须拥有一个可以处理 Amazon S3 重定向响应的应用程序。唯一的例外是专用于处理存储桶（在不使用 `<CreateBucketConfiguration>` 的情况下创建）的应用程序。有关存储桶位置约束的更多信息，请参阅访问和列出 Amazon S3 存储桶。

对于在 2019 年 3 月 20 日发布的所有区域，如果请求到达错误的 Amazon S3 位置，则 Amazon S3 返回 HTTP 400 错误请求错误。
有关启用或禁用 AWS 区域的更多信息，请参阅《AWS 一般参考》中的 AWS 区域 和端点。

主题

DNS 路由选择

DNS 路由选择会将请求路由到合适的 Amazon S3 设备。下面的图和过程显示 DNS 路由选择的示例。



DNS 路由请求步骤

1. 客户端创建 DNS 请求来获取存储在 Amazon S3 上的对象。
2. 客户端收到可以处理请求的设备的一个或多个 IP 地址。在本示例中，IP 地址用于设备 B。

3. 客户端创建面向 Amazon S3 设备 B 的请求。
4. 设备 B 将对象的副本返回给客户端。

临时请求重定向

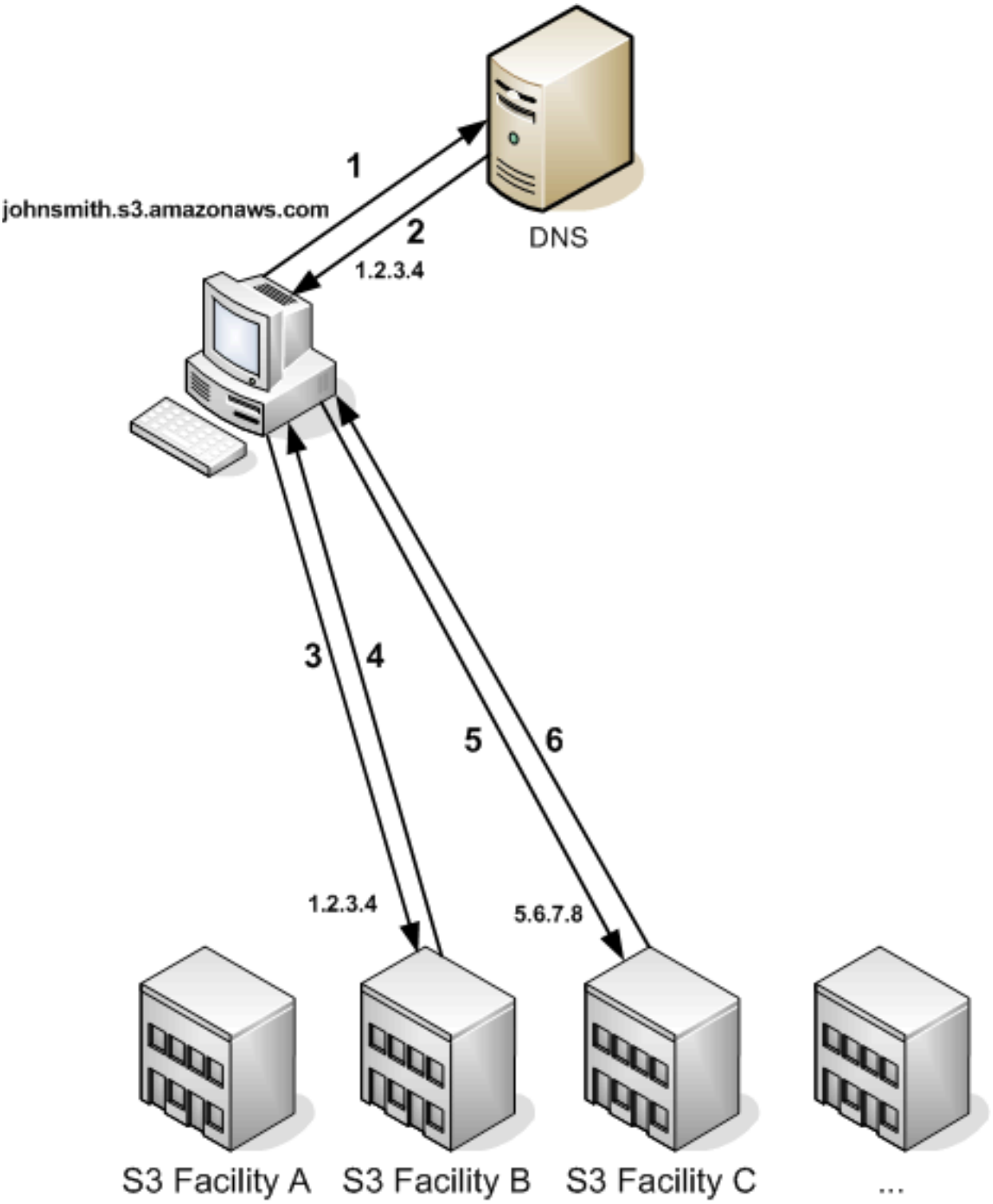
临时重定向是一种错误响应类型，指示请求者应将请求重新发送到其他终端节点。由于 Amazon S3 的分布式特性，请求可能暂时会被路由到错误的设备。创建或删除存储桶后，很可能会立即发生此情况。

例如，如果您创建新的存储桶并立即向该存储桶发送请求，则根据存储桶的位置约束，您可能收到临时重定向。如果您在美国东部（弗吉尼亚北部）AWS 区域 中创建了该存储桶，则看不到此重定向，因为这也是默认的 Amazon S3 终端节点。

但如果存储桶是在其他任何区域创建的，则对该存储桶的任何请求都会转到默认终端节点，同时会传播该存储桶的 DNS 条目。默认终端节点会将此请求重定向到具有 HTTP 302 响应的正确终端节点。临时重定向包含指向正确设备的 URI，您可以使用它来立即重新发送请求。

> **Important**
>
> 不要重复使用之前重定向响应提供的终端节点。它似乎能够正常工作（甚至可以持续很长一段时间），但它可能会产生不可预测的结果，且最终会在不另行通知的情况下导致失败。

下面的图和过程显示临时重定向的示例。



临时请求重定向步骤

1. 客户端创建 DNS 请求来获取存储在 Amazon S3 上的对象。
2. 客户端收到可以处理请求的设备的一个或多个 IP 地址。

3. 客户端创建面向 Amazon S3 设备 B 的请求。
4. 设备 B 将返回一个重定向，指示可从位置 C 获取对象。
5. 客户端会将请求重新发送到设备 C。
6. 设备 C 将返回对象的副本。

永久请求重定向

永久重定向指示请求中资源的寻址不正确。例如，如果您使用路径类型请求来访问使用 `<CreateBucketConfiguration>` 创建的存储桶，将发生永久重定向。有关更多信息，请参阅 访问和列出 Amazon S3 存储桶。

为帮助您在开发期间查找这些错误，此类型的重定向不包含位置 HTTP 标头 (该标头允许您自动遵循请求，以定向到正确的位置)。参考生成的 XML 错误文档，获取有关使用正确的 Amazon S3 终端节点的帮助。

请求重定向示例

以下是临时请求重定向响应的示例。

REST API 临时请求重定向

```
HTTP/1.1 307 Temporary Redirect
Location: http://awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com/photos/puppy.jpg?
rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint.
  Continue to use the original request endpoint for future requests.</Message>
  <Endpoint>awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com</Endpoint>
</Error>
```

SOAP API 临时请求重定向

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
    Continue to use the original request endpoint for future requests.</Faultstring>
    <Detail>
      <Bucket>images</Bucket>
      <Endpoint>s3-gztb4pa9sq.amazonaws.com</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

DNS 注意事项

Amazon S3 的设计要求之一是实现高可用性。为达到此要求，我们采取的一个方法是根据需要更新 DNS 中与 Amazon S3 终端节点相关联的 IP 地址。这些更改将自动在短期有效的客户端而不是某些长期有效的客户端中反映出来。长期有效的客户端将需要采取特殊的操作，定期重新解析 Amazon S3 终端节点，才能从这些更改中获益。有关虚拟机 (VM) 的更多信息，请参阅以下内容：

- 对于 Java，Sun 的 JVM 在默认情况下将永久缓存 DNS 查找结果；请参阅 InetAddress 文档中的“InetAddress 缓存”一节，以了解有关如何更改此行为的信息。
- 对于 PHP，在最常见的部署配置中运行的永久性 PHP VM 将缓存 DNS 查找结果，直到 VM 重新启动。请参阅 getHostByName PHP 文档。

处理 REST 和 SOAP 错误

主题

本节描述 REST 和 SOAP 错误，以及如何处理它们。

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

REST 错误响应

如果 REST 请求导致错误，则 HTTP 回复将包含：

- 作为响应正文的 XML 错误文档
- Content-Type: application/xml
- 合适的 3xx、4xx 或 5xx HTTP 状态代码

下面是 REST 错误响应的示例。

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

有关 Amazon S3 错误的更多信息，请参阅 ErrorCodeList。

响应标头

下面是所有操作返回的响应标头：

- `x-amz-request-id`：系统分配给每个请求的唯一 ID。在极少的情况下，如果您在使用 Amazon S3 时遇到问题，Amazon 将使用此信息来帮助排查问题。
- `x-amz-id-2`：将帮助我们排查问题的特殊令牌。

错误响应

若 Amazon S3 请求出现错误，客户端将收到一个错误响应。准确的错误响应格式应该特定于 API：例如，REST 错误响应的格式和 SOAP 错误响应的格式不同。但是，所有错误响应都有一些共同的元素。

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

错误代码

错误代码是用于唯一标识错误条件的字符串。这意味着按类型检测和处理错误的程序将读取和理解错误代码。许多错误代码在 SOAP 和 REST API 上都比较常见，但也有部分是特定于 API 的。例如，NoSuchKey 是通用的，但是 UnexpectedContent 仅在响应无效的 REST 请求时发生。在所有案例中，SOAP 错误代码会带有一个前缀 (如错误代码表格中所示)，以便 NoSuchKey 错误实际将作为 Client.NoSuchKey 返回到 SOAP 中。

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

错误消息

错误消息包含错误条件的常规描述 (英语)。它主要面向用户受众。如果用户遇到不知如何处理或不愿处理的错误条件，简单的程序将直接向最终用户显示消息。支持更详尽的错误处理和适当国际化的复杂程序更容易忽略错误消息。

更多详细信息

许多错误响应包含额外的结构化数据，旨在供开发人员阅读和理解以诊断编程错误。例如，如果您使用与服务器上计算出的摘要不匹配的 REST PUT 请求发送 Content-MD5 标头，您将收到 BadDigest 错误。错误响应还包括我们计算出的摘要的详细元素，以及预期内容的摘要。在开发过程中，您可以使用此信息诊断错误。在生产过程中，运行良好的程序可能会将此信息包含在其错误日志中。

SOAP 错误响应

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

在 SOAP 中，错误结果将作为 SOAP 错误 (使用 HTTP 响应代码 500) 返回到客户端。若您没有收到 SOAP 错误，则您的请求已成功。Amazon S3 SOAP 错误代码是由标准 SOAP 1.1 错误代码 (“Server”或“Client”) 与特定于 Amazon S3 的错误代码相连接而组成的。例如，“Server.InternalError”或“Client.NoSuchBucket”。SOAP 错误字符串元素包括一个通用的、用户可读的错误消息 (英语)。最后，SOAP 错误详细信息元素将包括与错误相关的其他信息。

例如，如果您尝试删除对象“Fred”，但该对象不存在，则 SOAP 响应的正文将包含“NoSuchKey”SOAP 错误。

Example

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

有关 Amazon S3 错误的更多信息，请参阅 ErrorCodeList。

Amazon S3 排错最佳实践

设计使用 Amazon S3 的应用程序时，正确处理 Amazon S3 错误非常重要。本节描述了设计应用程序时，您应该考虑的问题。

重试 InternalErrors

内部错误是指在 Amazon S3 环境中发生的错误。

可能还没有处理收到 InternalError 响应的请求。例如，如果 PUT 请求返回 InternalError，则后续的 GET 操作可能会检索旧的值或更新的值。

如果 Amazon S3 返回 InternalError 响应，请重新提交请求。

针对重复的 SlowDown 错误调整应用程序

与其他分布式系统一样，S3 的保护机制能够检测出有意或无意的资源过度消耗，并做出相应的反应。在较高的请求速率触发了其中某个保护机制后，将发生 SlowDown 错误。降低您的请求速率将减少或消除此类型的错误。一般而言，大多数用户不会经常遇到这些错误；但是，如果您希望了解更多信息，或遇到了严重或意外的 SlowDown 错误，请将该错误发布到我们的 Amazon S3 开发人员论坛或在以下网址注册 AWS Support：(https://aws.amazon.com/premiumsupport/)。

隔离错误

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

Amazon S3 提供了一组由 SOAP 和 REST API 共同使用的错误代码。SOAP API 将返回标准的 Amazon S3 错误代码。REST API 的设计与标准的 HTTP 服务器相似，它与现有的 HTTP 客户端 (例如，浏览器、HTTP 客户端库、代理、缓存等等) 进行交互。为了确保 HTTP 客户端能够正确地处理错误，我们将每个 Amazon S3 错误都映射为一个 HTTP 状态代码。

HTTP 状态代码没有 Amazon S3 错误代码那样直观，并且包含的错误信息也较少。例如，`NoSuchKey` 和 `NoSuchBucket` Amazon S3 错误均映射为 `HTTP 404 Not Found` 状态代码。

尽管 HTTP 状态代码包含较少的错误信息，但能够理解 HTTP 而不是 Amazon S3 API 的客户端通常能够正确地处理错误。

因此，处理错误或向最终用户报告 Amazon S3 错误时，请使用 Amazon S3 错误代码而不是 HTTP 状态代码，这是因为前者包含最详细的错误信息。此外，调试应用程序时，您还应该参考 XML 错误响应的人类可读的 <Details> 元素。

开发者参考

此附录包括下列部分。

主题

附录 A：使用 SOAP API

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 特征。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

本部分包含 Amazon S3 SOAP API 的特定信息。

> **Note**
>
> 必须使用 SSL 将经身份验证的和匿名的 SOAP 请求发送到 Amazon S3。若您通过 HTTP 发送 SOAP 请求，Amazon S3 将返回错误。

主题

常见 SOAP API 元素

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

您可以使用 SOAP 1.1 通过 HTTP 与 Amazon S3 进行交互。Amazon S3 WSDL 以机器可读的方式描述了 Amazon S3 API，可在以下网址获得：https://doc.s3.amazonaws.com/2006-03-01/

AmazonS3.wsdl。Amazon S3 架构可在 https://doc.s3.amazonaws.com/2006-03-01/AmazonS3.xsd 上找到。

大多数用户将使用为他们的语言和开发环境定制的 SOAP 工具包与 Amazon S3 进行交互。不同的工具包将以不同的方式提供 Amazon S3 API。请参照您的特定工具包文档来了解如何使用它。本节通过在 Amazon S3 SOAP 操作显示“在线”时呈现 XML 请求和响应，以独立于工具包的方式说明这些操作。

常见元素

您可以包含以下具有任何 SOAP 请求的与授权相关的元素：

- `AWSAccessKeyId`: 请求者的 AWS 访问密钥 ID
- `Timestamp`:系统上的当前时间
- `Signature`:用于该请求的签名

对 SOAP 请求进行身份验证

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 功能。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

所有非匿名的请求都必须包含身份验证信息，以便确立提出请求的委托人的身份。在 SOAP 中，身份验证信息放置在 SOAP 请求的以下元素中：

- 您的 AWS 访问密钥 ID

  > **Note**
  >
  > 创建经身份验证的 SOAP 请求时，不支持临时的安全凭证。有关凭证类型的更多信息，请参阅 提出请求。

- `Timestamp`: 它必须采用协调世界时 (Greenwich Mean Time) 时区的日期时间 (转到 http://www.w3.org/TR/xmlschema-2/#dateTime)，例如 `2009-01-01T12:00:00.000Z`。如果这个时间戳与 Amazon S3 服务器上的时间相差 15 分钟以上，授权将失败。

- `Signature:`“AmazonS3” + OPERATION + Timestamp 串联的 RFC 2104 HMAC-SHA1 摘要（转至 http://www.ietf.org/rfc/rfc2104.txt）将您的 AWS 秘密访问密钥用作密钥。例如，在下面的 CreateBucket 示例请求中，签名元素将包含“AmazonS3CreateBucket2009-01-01T12:00:00.000Z”值的 HMAC-SHA1 摘要：

例如，在下面的 CreateBucket 示例请求中，签名元素将包含“AmazonS3CreateBucket2009-01-01T12:00:00.000Z”值的 HMAC-SHA1 摘要：

Example

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-01-01T12:00:00.000Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

> **Note**
>
> 必须使用 SSL 将经身份验证的和匿名的 SOAP 请求发送到 Amazon S3。若您通过 HTTP 发送 SOAP 请求，Amazon S3 将返回错误。

> **Important**
>
> 由于对如何丢弃额外的时间精度存在不同的解释，.NET 用户应注意不要使用过于具体的时间戳来发送 Amazon S3。可以通过手动构建只有毫秒精度的 `DateTime` 对象完成此操作。

使用 SOAP 设置访问策略

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 特征。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

在编写存储桶或对象时，可以通过将“AccessControlList”元素随附在 CreateBucket、PutObjectInline 或 PutObject 的请求中，设置访问控制。AccessControlList 元素在 Amazon S3 的身份和访问管理 中进行了描述。如果没有使用这些操作指定访问控制列表，使用默认访问策略创建的资源将给予请求者 FULL_CONTROL 访问权限 (即使请求是用于已存在的对象的 PutObjectInline 或 PutObject 请求)。

下面是向对象写入数据、允许匿名委托人读取对象，以及授予特定用户对存储桶的 FULL_CONTROL 权限 (大多数开发人员希望授予他们自己对存储桶的 FULL_CONTROL 访问权限) 的请求。

Example

下面是向对象写入数据并允许匿名委托人读取对象的请求。

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot828ef3fdfa96f00ad9f27c383fc9ac7f&quot</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

可以使用 GetBucketAccessControlPolicy、GetObjectAccessControlPolicy、SetBucketAccessContro 和 SetObjectAccessControlPolicy 方法为现有存储桶或对象读取或设置访问控制策略。有关详细信息，请参阅这些方法的详细说明。

附录 B：对请求进行身份验证 (AWS Signature Version 2)

> **Important**
>
> 此部分介绍如何使用 AWS Signature Version 2 对请求进行身份验证。Signature Version 2 会关闭（弃用），Amazon S3 将仅接受使用 Signature Version 4 进行签名的 API 请求。有关更多信息，请参阅 AWS已为 Amazon S3 关闭（弃用）Signature Version 2。
> 所有 AWS 区域 都支持 Signature Version 4，这是新区域唯一支持的版本。有关更多信息，请参阅《Amazon Simple Storage Service API》参考中的验证请求 (AWS Signature Version 4)。
> Amazon S3 使您能够确定用于签署请求的 API 签名版本。请务必确定您的任何工作流是否正在使用 Signature Version 2 签名并将其升级到使用 Signature Version 4 以防影响您的业务。
>
> - 如果您使用的是 CloudTrail 事件日志（推荐选项），请参阅使用 CloudTrail 识别 Amazon S3 签名版本 2 请求以了解如何查询和确定此类请求。
> - 如果您使用的是 Amazon S3 服务器访问日志，请参阅使用 Amazon S3 访问日志确定签名版本 2 请求。

主题

- 使用 REST API 对请求进行身份验证
- 签署和对 REST 请求进行身份验证
- 使用 POST (AWS Signature Version 2) 的基于浏览器的上传

使用 REST API 对请求进行身份验证

使用 REST 访问 Amazon S3 时，您必须在请求中提供以下项目，以便对请求进行身份验证：

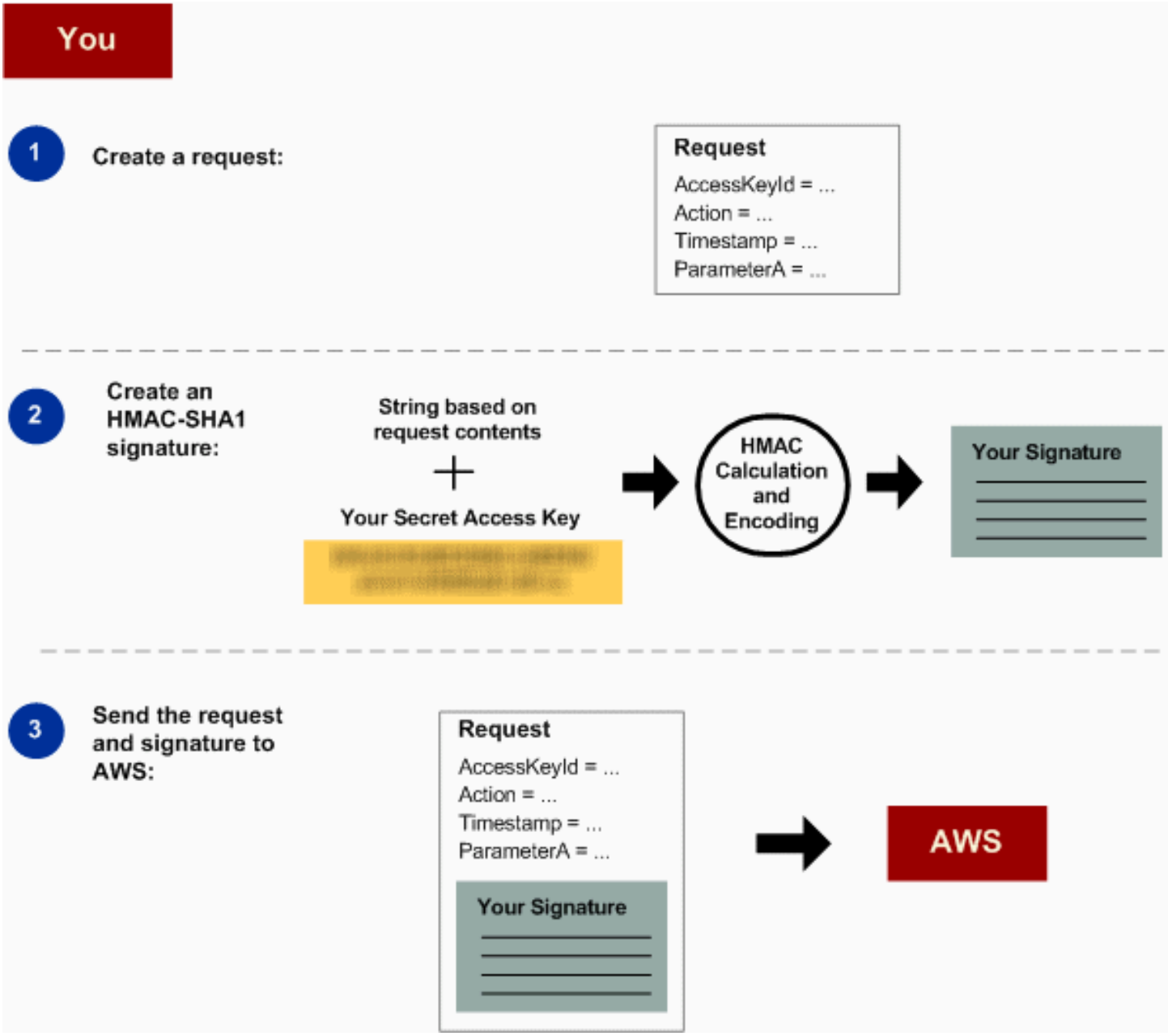
请求元素

- AWS 访问密钥 ID – 每个请求必须包含用于发送请求的身份的访问密钥 ID。
- 签名 - 每个请求必须包含一个有效的请求签名，否则系统会拒绝请求。

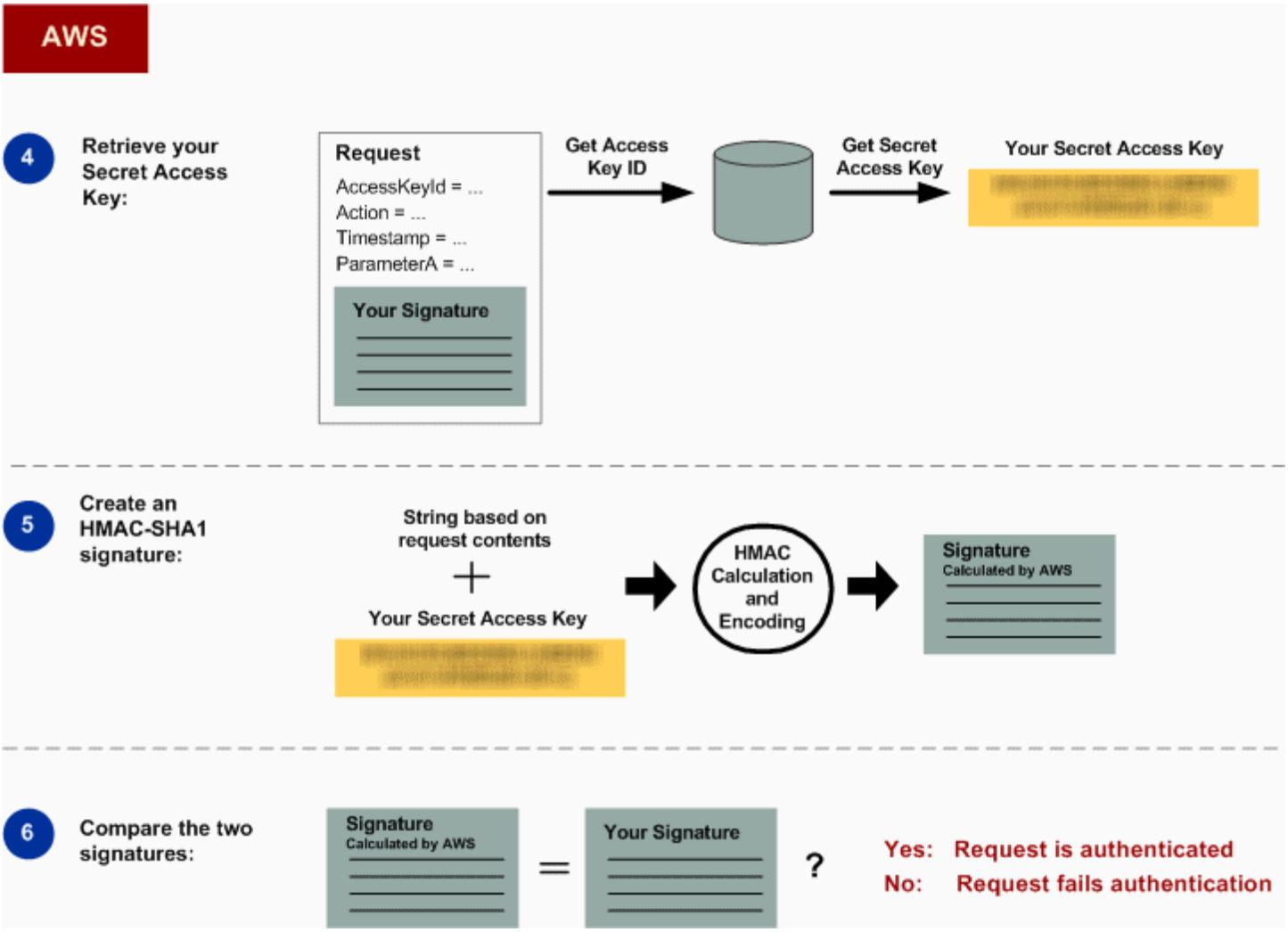
  将使用您的秘密访问密钥计算出请求签名，该密钥是一个共享密钥，相关知晓方只有您和 AWS。
- 时间戳 – 每个请求必须包含请求的创建日期和时间，并表示为 UTC 形式的字符串。
- 日期 – 每个请求都必须包含请求的时间戳。

  根据您正在使用的 API 操作，您可以不提供时间戳，改为提供请求的过期日期和时间，或者也可以二者都提供。要确定特定操作需要的内容，请参阅该操作的身份验证主题。

下面是针对发送到 Amazon S3 的请求进行身份验证的常规步骤。此处假设您已拥有了必要的安全凭证、访问密钥 ID 和秘密访问密钥。



1	构建要发送到 AWS 的请求。
2	使用您的秘密访问密钥计算签名。
3	向 Amazon S3 发送请求。在请求中包含您的访问密钥 ID 和签名。Amazon S3 执行下面三个步骤。



4	Amazon S3 使用访问密钥 ID 查找秘密访问密钥。
5	通过与用于计算您在请求中发送的签名相同的算法，Amazon S3 可根据请求数据和秘密访问密钥计算出签名。
6	如果由 Amazon S3 生成的签名与您在请求中发送的签名相匹配，将认为请求是真实的。如果比较签名这一操作失败，那么系统会丢弃请求，同时 Amazon S3 将返回错误响应。

详细的身份验证信息

有关 REST 身份验证的详细信息，请参阅 签署和对 REST 请求进行身份验证。

签署和对 REST 请求进行身份验证

主题

> **Note**
>
> 本主题说明使用签名版本 2 对请求进行身份验证。Amazon S3 现在支持最新的 Signature Version 4。所有区域都支持此最新签名版本，而 2014 年 1 月 30 日之后的所有新区域都只支持签名版本 4。有关更多信息，请转至《Amazon Simple Storage Service API 参考》中的验证请求 (AWS Signature Version 4)。

身份验证是指向系统证明身份的过程。身份对于 Amazon S3 访问控制决策非常重要。将根据请求者的身份允许请求或拒绝部分请求。例如，将为已注册的开发人员保留创建存储桶的权利，并且 (默认) 将为相关的存储桶拥有者保留在存储桶中创建对象的权利。作为开发人员，您需要提出请求来调用这些权限，因此您需要对您的请求进行身份验证以向系统证明您的身份。本节向您演示了应如何进行操作。

> **Note**
>
> 本节中的内容不适用于 HTTP POST。有关更多信息，请参阅 使用 POST (AWS Signature Version 2) 的基于浏览器的上传。

Amazon S3 REST API 使用基于密钥 HMAC (Hash Message Authentication Code) 的自定义 HTTP 方案进行身份验证。要对请求进行身份验证，您首先需要合并请求的选定元素以形成一个字符串。然后，

您可以使用 AWS 秘密访问密钥来计算该字符串的 HMAC。通常我们将此过程称为“签署请求”并且我们将输出 HMAC 算法称为“签名”，因为它会模拟真实签名的安全属性。最后，您可以使用本部分中介绍的语法，作为请求的参数添加此签名。

系统收到经身份验证的请求时，将提取您申领的 AWS 秘密访问密钥，并以相同的使用方式将它用于计算已收到的消息的签名。然后，它会将计算出的签名与请求者提供的签名进行对比。如果两个签名相匹配，则系统认为请求者必须拥有对 AWS 秘密访问密钥的访问权限，因此充当向其颁发密钥的委托人的颁发机构。如果两个签名不匹配，那么请求将被丢弃，同时系统将返回错误消息。

Example 经身份验证的 Amazon S3 REST 请求

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket1.us-west-1.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000

Authorization: AWS AKIAIOSFODNN7EXAMPLE:
qgk2+6Sv9/oM7G3qLEjTH1a1l1g=
```

使用临时安全凭证

如果您使用临时安全凭证签署请求 (参阅 提出请求)，您必须通过添加 `x-amz-security-token` 标头将相应的安全令牌包含在您的请求中。

使用 AWS Security Token Service API 获取临时安全凭证时，响应包含临时安全凭证和会话令牌。向 Amazon S3 发送请求时，您可以在 `x-amz-security-token` 标头中提供会话令牌值。有关 IAM 提供的 AWS Security Token Service API 的信息，请转到《AWS Security Token Service API 参考指南》中的操作。

身份验证标头

Amazon S3 REST API 使用标准的 HTTP `Authorization` 标头来传递身份验证信息。（标准标头的名称是不可取的，因为它承载的是身份验证信息，而不是授权。）在 Amazon S3 身份验证方案下，授权标头具有以下形式：

```
Authorization: AWS AWSAccessKeyId:Signature
```

注册后，会向开发人员发放 AWS 访问密钥 ID 和 AWS 秘密访问密钥。对于请求身份验证，`AWSAccessKeyId` 元素将标识用于计算签名的访问密钥 ID 和进行请求的开发人员 (间接)。

Signature 元素是请求中选定元素的 RFC 2104HMAC-SHA1，因此授权标头的 Signature 部分会因请求的不同而各异。如果系统计算出的请求签名与请求随附的 Signature 相匹配，则请求者拥有经证明的 AWS 秘密访问密钥。然后，在该身份下，借助获得此密钥的开发人员的授权来处理请求。

以下是说明 Authorization 请求标头结构的伪语法。(在该示例中，\n 表示 Unicode 码位 U +000A，这通常称为换行符)。

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature;

Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-
Of( StringToSign ) ) );

StringToSign = HTTP-Verb + "\n" +
 Content-MD5 + "\n" +
 Content-Type + "\n" +
 Date + "\n" +
 CanonicalizedAmzHeaders +
 CanonicalizedResource;

CanonicalizedResource = [ "/" + Bucket ] +
 <HTTP-Request-URI, from the protocol name up to the query string> +
 [ subresource, if present. For example "?acl", "?location", or "?logging"];

CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 是由 RFC 2104（用于消息身份验证的哈希密钥）定义的算法。该算法要求输入两个字节字符串：一个密钥和一个消息。对于 Amazon S3 请求身份验证，请将您的 AWS 秘密访问密钥 (YourSecretAccessKey) 用作密钥，并将 UTF-8 编码格式的 StringToSign 用作消息。HMAC-SHA1 的输出也是字节字符串，称为摘要。通过对此摘要进行 Base64 编码来构成 Signature 请求参数。

用于签名的请求规范

在系统收到经身份验证的请求时撤销，系统会将计算出的请求签名与 StringToSign 中提供的签名进行对比。基于此原因，您必须采用 Amazon S3 使用的相同方法计算签名。根据适用于签名的统一格式放置请求的过程称为标准化。

构建 CanonicalizedResource 元素

CanonicalizedResource 表示请求的目标 Amazon S3 资源。为 REST 请求构建它，如下所示：

启动过程

1	使用空字符串 ("") 启动。
2	如果请求使用 HTTP 主机标头 (虚拟托管类型) 来指定请求，请在 "/" (例如，“/bucketname”) 的后面附加存储桶名称。对于路径类型请求和不会寻址存储桶的请求，不执行任何操作。有关虚拟托管类型请求的更多信息，请参阅存储桶的虚拟托管。 对于虚拟托管类型请求“https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg”，CanonicalizedResource 是“/awsexamplebucket1”。 对于路径类型请求“https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg”，CanonicalizedResource 为 ""。
3	附加未解码的 HTTP 请求-URI 的路径部分 (取决于但不包括查询字符串)。 对于虚拟托管类型请求“https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg”，CanonicalizedResource 是“/awsexamplebucket1/photos/puppy.jpg”。 对于路径类型的请求，“https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg”，CanonicalizedResource 是“/awsexamplebucket1/photos/puppy.jpg”。此时，虚拟托管类型和路径类型请求的 CanonicalizedResource 相同。 对于不寻址存储桶的请求（例如，GET Service），请附加“/”。
4	如果请求将寻址子资源（例如，?versioning 、?location 、?acl、?lifecycle 或 ?versionid ），请附加子资源、其值（如果有）和问号。请注意，如果存在多个子资源，子资源必须按子资源名称的字典顺序排序并使用“&”进行分隔 (例如，?acl&versionId=#)。 构建 CanonicalizedResource 元素时必须包含的子资源为：acl、lifecycle、location、logging、notification、partNumber、policy、requestPayment、uploadId、uploads、versionId、versioning、versions 和 website。 如果请求指定覆盖响应标头值的查询字符串参数（请参阅 Get Object），请附加查询字符串参数及其值。签名时，请勿对这些值进行编码；但是进行请求时，您必须对这些参数值进行编码。GET 请求中的查询字符串参数包括 response-content-type 、response-content-language 、response-expires 、response-cache-control 、response-content-disposition 和 response-content-encoding 。

	为多对象删除请求创建 CanonicalizedResource 时，必须包括 delete 查询字符串参数。

必须按照元素在 HTTP 请求中的显示方式，使用字母 (包括 URL 编码元字符) 签署来自 HTTP 请求-URI 的 CanonicalizedResource 元素。

CanonicalizedResource 可能与 HTTP 请求-URI 不同。尤其是，如果您的请求使用 HTTP Host 标头来指定存储桶，存储桶不会在 HTTP 请求 URI 中显示。但是，CanonicalizedResource 仍然会包含该存储桶。查询字符串参数可能也会在请求-URI 中显示，但是不会包括在 CanonicalizedResource 中。有关更多信息，请参阅 存储桶的虚拟托管。

构建 CanonicalizedAmzHeaders 元素

要构建 StringToSign 的 CanonicalizedAmzHeaders 部分，请选择所有以“x-amz-”开头的 HTTP 请求标头 (使用不区分大小写的对比方式) 并遵循以下步骤。

CanonicalizedAmzHeaders 步骤

1	将每个 HTTP 标头名称转换为小写。例如，“X-Amz-Date ”改为“x-amz-date ”。
2	根据标头名称按字典顺序排列标头集。
3	将相同名称的标头字段合并为一个“header-name:comma-separated-value-list”对，并按照 RFC 2616 中第 4.2 节中的规定，两个值之间不留空格。例如，可以将元数据标头“x-amz-meta-username: fred ”和“x-amz-meta-username: barney ”合并为单个标头“x-amz-meta-username: fred,barney ”。
4	通过将折叠空格（包括新建行）替换为单个空格，“展开”跨多个行的长标头（按照 RFC 2616 中第 4.2 节允许的方式）。
5	删除标头中冒号周围的空格。例如，标头“x-amz-meta-username: fred,barney ”改为“x-amz-meta-username:fred,barney ”。
6	最后，请向生成的列表中的每个标准化标头附加换行字符 (U+000A)。通过将此列表中所有的标头规范化为单个字符串，构建 CanonicalizedResource 元素。

位置和命名 HTTP 标头 StringToSign 元素的对比

StringToSign 的头几个标头元素（Content-Type、Date 和 Content-MD5）属于位置标头。StringToSign 不包括这些标头的名称，仅包括它们在请求中的值。相反，“x-amz-”元素会进行命名。标头名称和标头值都会在 StringToSign 中显示。

如果在 StringToSign 定义中调用的位置标头不在请求中 (例如，Content-Type 或 Content-MD5 对于 PUT 请求是可选的，并且对于 GET 请求没有任何意义)，请使用空字符串 ("") 替换该位置。

时间戳要求

有效的时间戳对于经身份验证的请求是必须的 (使用 HTTP Date 标头或 x-amz-date 替代项)。此外，经身份验证的请求随附的客户端时间戳必须处于收到请求时的 Amazon S3 系统时间的 15 分钟之内。否则，请求将失败并出现 RequestTimeTooSkewed 错误代码。施加这些限制的目的是为了防止对方重新使用已拦截的请求。要更好地防范窃听，请对经身份验证的请求使用 HTTPS 传输。

> **Note**
>
> 对请求日期的验证限制仅适用于不使用查询字符串身份验证的经身份验证的请求。有关更多信息，请参阅 查询字符串请求身份验证替代项。

某些 HTTP 客户端库不提供为请求设置 Date 标头的功能。如果您在标准化标头中包含“Date”标头的值时遇到困难，您可以改用“x-amz-date”标头为请求设置时间戳。x-amz-date 标头的值必须采用其中一种 RFC 2616 格式 (http://www.ietf.org/rfc/rfc2616.txt)。x-amz-date 标头位于请求中时，系统将在计算请求签名时忽略任何 Date 标头。因此，如果包含了 x-amz-date 标头，请在构建 Date 时使用 StringToSign 的空字符串。有关示例，请参阅下一节。

身份验证示例

本节中的示例使用下表中的 (非有效) 凭证。

参数	值
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecret AccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

在示例 StringToSign 中，格式并不重要；\n 是指 Unicode 码位 U+000A，通常称为换行符。此外，该示例使用“+0000”指定时区。您可以改用“GMT”来指定时区，但是在示例中显示的签名将有所不同。

对象 GET

此示例将从 awsexamplebucket1 存储桶获取一个对象。

请求	StringToSign
`GET /photos/puppy.jpg HTTP/1.1` `Host: awsexamplebucket1.us-west-1.s3.amazonaws.com` `Date: Tue, 27 Mar 2007 19:36:42 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:qgk2+6Sv9/oM7G3qLEjTH1a1l1g=`	`GET\n` `\n` `\n` `Tue, 27 Mar 2007 19:36:42 +0000\n` `/awsexamplebucket1/photos/puppy.jpg`

请注意，CanonicalizedResource 包括存储桶名称，但 HTTP 请求 URI 不包括。（存储桶由主机标头指定）。

> **Note**
>
> 以下 Python 脚本使用提供的参数来计算前面的签名。您可以使用此脚本来构建您自己的签名，并根据需要替换密钥和 StringToSign。

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSFODNN7EXAMPLE'.encode("UTF-8")
secret_key = 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/photos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
                                hmac.new(
                                        secret_key, string_to_sign, sha1
```

```
                                        ).digest()
                                ).strip()


print(f"AWS {access_key.decode()}:{signature.decode()}")
```

对象 PUT

此示例将向 awsexamplebucket1 存储桶放入一个对象。

请求	StringToSign
`PUT /photos/puppy.jpg HTTP/1.1` `Content-Type: image/jpeg` `Content-Length: 94328` `Host: awsexamplebucket1.s3.us-wes` `t-1.amazonaws.com` `Date: Tue, 27 Mar 2007 21:15:45 +0000` *`Authorization: AWS AKIAIOSFODNN7EXAMP`* *`LE:`* *`iqRzw+ileNPu1fhspnRs8nOjjIA=`*	`PUT\n` `\n` `image/jpeg\n` `Tue, 27 Mar 2007 21:15:45 +0000\n` `/awsexamplebucket1/photos/puppy.jpg`

请注意请求和 StringToSign 中的 Content-Type 标头。另请注意，由于 Content-MD5 不在请求中，它在 StringToSign 中将保留为空白。

列出

此示例将列出 awsexamplebucket1 存储桶的内容。

请求	StringToSign
`GET /?prefix=photos&max-keys=50&marker=puppy` ` HTTP/1.1` `User-Agent: Mozilla/5.0` `Host: awsexamplebucket1.s3.us-west-1.amazo` `naws.com`	`GET\n` `\n` `\n` `Tue, 27 Mar 2007 19:42:41` ` +0000\n`

请求	StringToSign
`Date: Tue, 27 Mar 2007 19:42:41 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:` `m0WP8eCtspQl5Ahe6L1SozdX9YA=`	`/awsexamplebucket1/`

请注意 CanonicalizedResource 上的尾部斜杠以及查询字符串参数的缺失。

取回

本示例将为“awsexamplebucket1”存储桶提取访问控制策略子资源。

请求	StringToSign
`GET /?acl HTTP/1.1` `Host: awsexamplebucket1.s3.us-west-1.amazo` `naws.com` `Date: Tue, 27 Mar 2007 19:44:46 +0000` `Authorization: AWS AKIAIOSFODNN7EXAMPLE:` `82ZHiFIjc+WbcwFKGUVEQspPn+0=`	`GET\n` `\n` `\n` `Tue, 27 Mar 2007 19:44:46` `+0000\n` `/awsexamplebucket1/?acl`

请注意子资源查询字符串参数包含在 CanonicalizedResource 中的方式。

删除

本示例使用路径类型和日期替代项从“awsexamplebucket1”存储桶删除对象。

请求	StringToSign
`DELETE /awsexamplebucket1/photos/p` `uppy.jpg HTTP/1.1` `User-Agent: dotnet` `Host: s3.us-west-1.amazonaws.com` `Date: Tue, 27 Mar 2007 21:20:27 +0000` `x-amz-date: Tue, 27 Mar 2007 21:20:26` `+0000`	`DELETE\n` `\n` `\n` `Tue, 27 Mar 2007 21:20:26 +0000\n` `/awsexamplebucket1/photos/puppy.jpg`

请求	StringToSign
Authorization: AWS AKIAIOSFODNN7EXAMP LE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=	

请注意我们如何使用“x-amz-date”替代方法来指定日期 (假设是因为我们的客户端库阻止我们设置日期)。在这种情况下，x-amz-date 优先于 Date 标头。因此，签名中的日期条目必须包含 x-amz-date 标头的值。

上传

本示例将对象上传到使用元数据的别名记录虚拟托管类型存储桶。

请求	StringToSign
PUT /db-backup.dat.gz HTTP/1.1 User-Agent: curl/7.15.5 Host: static.example.com:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000 x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw== X-Amz-Meta-ReviewedBy: joe@example.com X-Amz-Meta-ReviewedBy: jane@exam ple.com X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; filename=database.dat Content-Encoding: gzip Content-Length: 5913339 *Authorization: AWS AKIAIOSFODNN7EXAMP LE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=*	PUT\n 4gJE4saaMU4BqNR0kLY+lw==\n application/x-download\n Tue, 27 Mar 2007 21:06:08 +0000\n x-amz-acl:public-read\n x-amz-meta-checksumalgorithm:c rc32\n x-amz-meta-filechecksum:0x026 61779\n x-amz-meta-reviewedby: joe@example.com,jane@example.com \n /static.example.com/db-backup.dat .gz

请注意对“x-amz-”标头排序、删减多余空格和转换为小写的方式。另请注意，具有相同名称的多个标头使用逗号连接，以分隔值。

请注意，如何仅使 Content-Type 和 Content-MD5 HTTP 实体标头显示在 StringToSign 中。其他 Content-* 实体标头不显示。

同样请注意，CanonicalizedResource 包括存储桶名称，但 HTTP 请求 URI 不包括。（存储桶由主机标头指定）。

列出我的所有存储桶

请求	StringToSign
`GET / HTTP/1.1` `Host: s3.us-west-1.amazonaws.com` `Date: Wed, 28 Mar 2007 01:29:59 +0000` *`Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdERIC03wnaRNKh6OqZehG9s=`*	`GET\n` `\n` `\n` `Wed, 28 Mar 2007 01:29:59 +0000\n` `/`

Unicode 密钥

请求	StringToSign
`GET /dictionary/fran%C3%A7ais/pr%c3%a9f%c3%a8re HTTP/1.1` `Host: s3.us-west-1.amazonaws.com` `Date: Wed, 28 Mar 2007 01:49:49 +0000` *`Authorization: AWS AKIAIOSFODNN7EXAMPLE:DNEZGsoieTZ92F3bUfSPQcbGmlM=`*	`GET\n` `\n` `\n` `Wed, 28 Mar 2007 01:49:49 +0000\n` `/dictionary/fran%C3%A7ais/pr%c3%a9f%c3%a8re`

> **Note**
>
> 将逐字读取来自请求-URI 的 StringToSign 中的元素，包括 URL 编码和大写。

REST 请求签名问题

REST 请求身份验证失败后，系统将使用 XML 错误文档对请求做出响应。此错误文档中包含的信息将帮助开发人员诊断问题。特别是，StringToSign 错误文档的 SignatureDoesNotMatch 元素将明确告诉您系统使用的请求规范。

某些工具包可能会在您事先不知情的情况下自动插入标头，例如在 PUT 操作期间添加标头 Content-Type。在大部分情况下，插入标头的值保持不变，从而使您可以使用诸如 Ethereal 或 tcpmon 等工具来发现丢失的标头。

查询字符串请求身份验证替代项

您可以通过传递请求信息作为查询字符串参数，而不是使用 Authorization HTTP 标头来验证特定类型的请求。这在允许第三方浏览器直接访问您的私有 Amazon S3 数据，而无需代理请求时非常有用。其概念是构建一个“预签名”的请求并将其编码为最终用户浏览器可检索的 URL。此外，您还可以通过指定过期时间来限制预签名请求。

有关使用查询参数对请求进行身份验证的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的对请求进行身份验证：使用查询参数 (AWS Signature Version 4)。有关使用 AWS 开发工具包生成预签名 URL 的示例，请参阅 使用预签名 URL 共享对象。

创建签名

以下是查询字符串验证的 Amazon S3 REST 请求示例。

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa
%2ByT272YEAiv4%3D HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000
```

查询字符串请求身份验证方法不要求任何特殊的 HTTP 标头，而是将要求的身份验证元素指定为查询字符串参数：

查询字符串参数名称	示例值	说明
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	您的 AWS 访问密钥 ID。指定用于签署请求的 AWS 秘密访问密钥以及（间接）进行请求的开发人员的身份。

查询字符串参数名称	示例值	说明
Expires	1141889120	将签名过期时间指定为自 Epoch (1970 年 1 月 1 日 00:00:00 UTC) 以来的秒数。将拒绝在此时间 (根据服务器) 之后收到的请求。
Signature	vjbyPxybdZaNmGa%2B yT272YEAiv4%3D	采用 StringToSign 的 HMAC-SHA1 Base64 编码的 URL 编码。

查询字符串请求身份验证方法与普通的方法稍有差异，不同之处仅在于 Signature 请求参数和 StringToSign 元素的格式。下面的伪语法演示了查询字符串请求身份验证方法。

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-
Of( StringToSign ) ) ) );

StringToSign = HTTP-VERB + "\n" +
    Content-MD5 + "\n" +
    Content-Type + "\n" +
    Expires + "\n" +
    CanonicalizedAmzHeaders +
    CanonicalizedResource;
```

YourSecretAccessKey 是在您注册成为 Amazon Web Services 开发人员时，Amazon 分配给您的 AWS 秘密访问密钥 ID。请注意如何对 Signature 进行 URL 编码，以使其适合放置在查询字符串中。另请注意，在 StringToSign 中，HTTP Date 位置元素已替换为 Expires。CanonicalizedAmzHeaders 和 CanonicalizedResource 是相同的。

> **Note**
>
> 在查询字符串身份验证方法中，在计算要签署的字符串时，请勿使用 Date 或 x-amz-date request 标头。

查询字符串请求身份验证

请求	StringToSign
`GET /photos/puppy.jpg?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&` `    Signature=NpgCjnDzrM%2BWFzoENXmpNDUsSn8%3D&` `    Expires=1175139620 HTTP/1.1` `Host: awsexamplebucket1.s3.us-west-1.amazonaws.com`	`GET\n` `\n` `\n` `1175139620\n` `/awsexamplebucket1/photos/puppy.jpg`

我们假设，在浏览器创建 GET 请求时，它没有提供 Content-MD5 或 Content-Type 标头，也没有设置任何 x-amz- 标头，因此 `StringToSign` 的这些部分都保留为空白。

使用 Base64 编码

必须使用 Base64 编码 HMAC 请求签名。Base64 编码将签名转换为可附加到请求的简单 ASCII 字符串。如果要在 URI 中使用诸如加号 (+)、正斜杠 (/) 和等号 (=) 等可在签名中显示的字符，必须对它们进行编码。例如，如果身份验证代码包括一个加号 (+) 标志，请在请求中将其编码为 %2B。将正斜杠编码为 %2F，并将等号编码为 %3D。

有关 Base64 编码的示例，请参考 Amazon S3 身份验证示例。

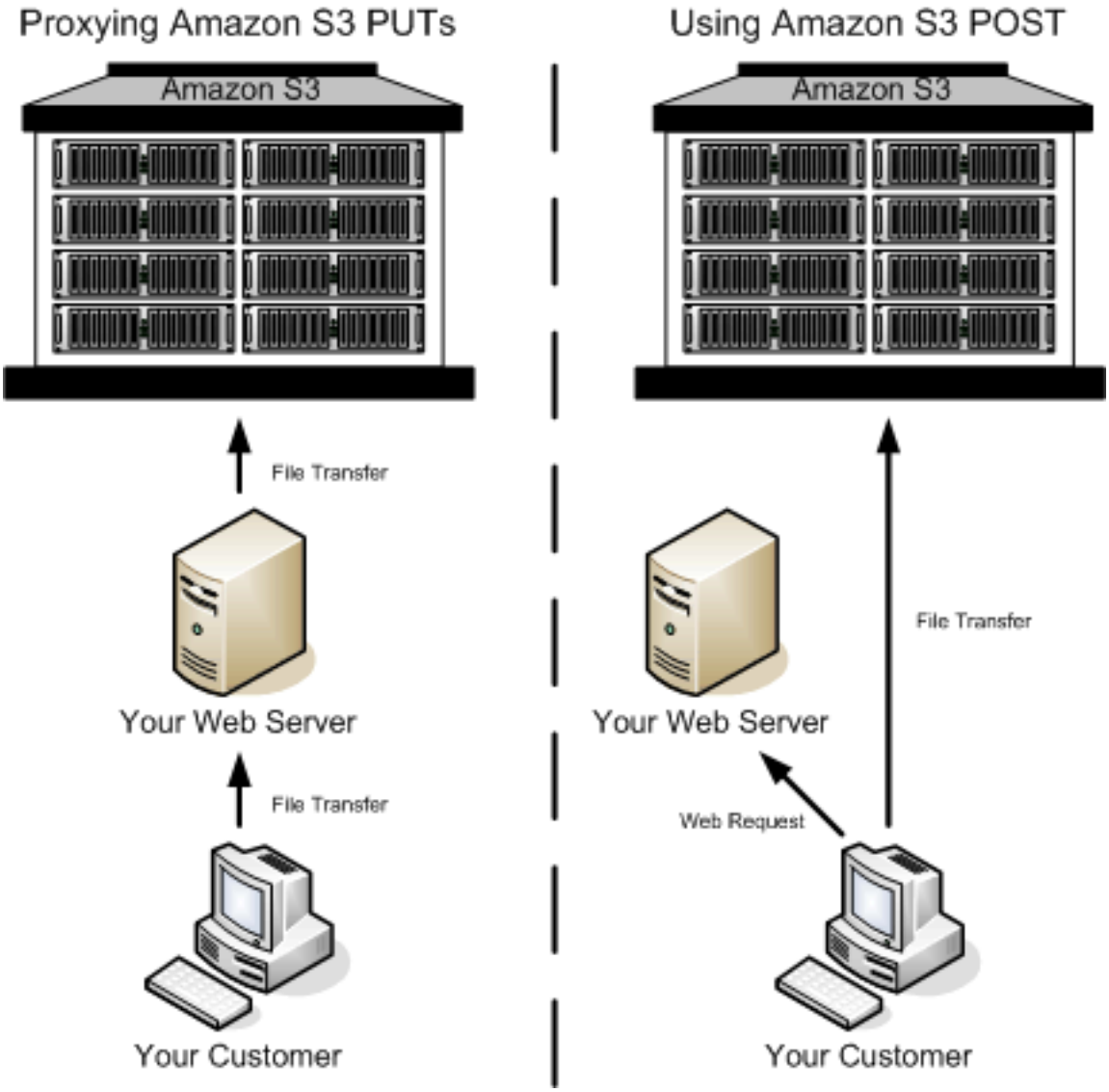
使用 POST (AWS Signature Version 2) 的基于浏览器的上传

Amazon S3 支持 POST，使您的用户可以直接将内容上传到 Amazon S3。POST 旨在简化上传过程和缩短上传延迟，而且可以节省用于上传数据以存储于 Amazon S3 中所用应用程序的开支。

> **Note**
>
> 本节中讨论的请求身份验证基于 AWS Signature Version 2，这是一种对 AWS 服务的入站 API 请求进行身份验证的协议。
> Amazon S3 现在在所有 AWS 区域 支持 Signature Version 4，后者是一种用于对 AWS 服务入站 API 请求进行身份验证的协议。目前，于 2014 年 1 月 30 日前创建的 AWS 区域 将继续支持之前的协议：Signature Version 2。于 2014 年 1 月 30 日后创建的所有新区域将只支持 Signature Version 4，因此，发往这些区域的所有请求都必须采用 Signature Version 4。

有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的使用 POST (AWS Signature Version 4) 在基于浏览器的上传中验证请求。

下图演示了使用 Amazon S3 POST 的上传。



使用 POST 上传

1	用户打开 Web 浏览器并访问您的 Web 页面。
2	Web 页面包含一个 HTTP 表格，其中包含了用户将内容上传到 Amazon S3 时必需的所有信息。
3	用户直接将内容上传到 Amazon S3。

> **Note**
>
> POST 不支持查询字符串身份验证。

HTML 表单 (AWS Signature Version 2)

主题

- HTML 表单编码
- HTML 表单声明
- HTML 表单字段
- 策略构建
- 构建签名
- 重定向

与 Amazon S3 进行通信时，通常可以使用 REST 或 SOAP API 来执行放置、获取、删除和其他操作。借助 POST，用户可通过其浏览器将数据直接上传到 Amazon S3，浏览器无法处理 SOAP API 或创建 REST PUT 请求。

> **Note**
>
> HTTP 上的 SOAP 支持已弃用，但 SOAP 仍可在 HTTPS 上使用。SOAP 不支持新增的 Amazon S3 特征。我们建议您使用 REST API 或 AWS SDK，而不是使用 SOAP。

要允许用户使用其浏览器将内容上传到 Amazon S3，要使用 HTML 表单。HTML 表单由表单声明和表单字段组成。表单声明包含关于请求的高级别信息。表单字段包含关于请求的详细信息，同时还包含用于对请求进行身份验证并确保其满足指定条件的策略。

> **Note**
>
> 表单数据和边界 (不包括文件的内容) 不得超过 20 KB。

本部分说明如何使用 HTML 表单。

HTML 表单编码

必须采用 UTF-8 编码表单和策略。您可以将 UTF-8 编码应用于表单，方法是在 HTML 标题中指定它或将它指定为请求标头。

> **Note**
>
> HTML 表单声明不接受查询字符串身份验证参数。

下面是 HTML 标题中 UTF-8 编码的示例：

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

下面是请求标头中 UTF-8 编码的示例：

```
Content-Type: text/html; charset=UTF-8
```

HTML 表单声明

表单声明包含三个部分：操作、方法和附件类型。如果这些值当中的任意一个设置不正确，请求将失败。

操作将指定处理请求的 URL，必须将它设置为存储桶的 URL。例如，如果您的存储桶名称为 awsexamplebucket1，区域为美国西部（加利福尼亚北部），则 URL 为 https://awsexamplebucket1.s3.us-west-1.amazonaws.com/。

> **Note**
>
> 在表单字段中指定键名。

方法必须是 POST。

必须为文件上传和文本区域上传都指定附件类型 (enctype) 并将其设置为“分段/表单数据”。有关更多信息，请转到 RFC 1867。

Example

以下示例是用于存储桶“awsexamplebucket1”的表单声明。

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"

enctype="multipart/form-data">
```

HTML 表单字段

下表介绍可以在 HTML 表单中使用的字段。

> **Note**
>
> 变量 `${filename}` 将自动替换为用户提供的文件的名称，并且可由所有表单字段识别。如果浏览器或客户端提供完整或部分文件路径，则只会使用跟在最后一个斜杠 (/) 或反斜杠 (\) 之后的文本。例如，“C:\Program Files\directory1\file.txt”会解释为“file.txt”。如果没有提供文件或文件名，变量将替换为空字符串。

字段名称	说明	必需
`AWSAccessKeyId`	存储桶拥有者的 AWS 访问密钥 ID，该拥有者授予匿名用户对满足策略中一组约束的请求的访问权限。如果请求包含策略文档，则此字段为必填字段。	条件
`acl`	Amazon S3 访问控制列表 (ACL)。如果指定了无效的访问控制列表，将产生错误。有关 ACL 的更多信息，请参阅 访问控制列表（ACL）。 类型：字符串 默认值：private	否

字段名称	说明	必需
	有效值：`private | public-read | public-read-write | aws-exec-read | authenticated-read | bucket-owner-read | bucket-owner-full-control`	
`Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires`	特定于 REST 的标头。有关更多信息，请参阅 PUT Object。	否
`key`	已上传的键的名称。 要使用由用户提供的文件名，请使用 ${filename} 变量。例如，如果用户 Betty 上传了文件 lolcatz.jpg 并且您指定了 /user/betty/${filename}，则该文件将存储为 /user/betty/lolcatz.jpg。 有关更多信息，请参阅 使用对象元数据。	是
`policy`	描述请求中允许的内容的安全策略。不带安全策略的请求被认为是匿名的，只会在公共可写的存储桶上成功。	否

字段名称	说明	必需
success_action_redirect, redirect	上传成功后客户端重定向到的 URL。Amazon S3 将存储桶、密钥和 etag 值作为查询字符串参数附加到 URL。 如果未指定 success_action_redirect，Amazon S3 将返回在 success_action_status 字段中指定的空文档类型。 如果 Amazon S3 无法解释 URL，它将忽略该字段。 如果上传失败，Amazon S3 将显示错误并且不会将用户重定向到某个 URL。 有关更多信息，请参阅重定向。 **Note** 已弃用重定向字段名称并且将在以后移除对重定向字段名称的支持。	否

字段名称	说明	必需
success_action_status	如果没有指定 success_action_redirect，上传成功后状态代码将返回到客户端。 有效值为 200、201 或 204 (默认)。 如果值设置为 200 或 204，Amazon S3 将返回一个空文档和一个 200 或 204 状态代码。 如果值设置为 201，Amazon S3 将返回一个 XML 文档和一个 201 状态代码。有关 XML 文档内容的信息，请参阅 POST Object。 如果没有设置值或者设置了无效的值，Amazon S3 将返回一个空文档和一个 204 状态代码。 **Note** 某些版本的 Adobe Flash player 无法正确处理使用空白正文的 HTTP 响应。要通过 Adobe Flash 支持上传，建议您将 success_action_status 设置为 201。	否
signature	HMAC 签名，使用与提供的 AWSAccessKeyId 对应的秘密访问密钥构建。如果策略文档没有随请求一起提供，则此字段为必填字段。 有关更多信息，请参阅 Amazon S3 的身份和访问管理。	条件

字段名称	说明	必需
x-amz-security-token	会话凭证使用的安全令牌 如果请求使用 Amazon DevPay，则它需要两个 x-amz-security-token 表单字段：一个用于产品令牌，另一个用于用户令牌。 如果请求使用会话凭证，则它需要一个 x-amz-security-token 表单。有关更多信息，请参阅《IAM 用户指南》中的临时安全凭证。	否
其他以 x-amz-meta- 为前缀的字段名称	特定于用户的元数据。 Amazon S3 没有验证或使用此数据。 有关更多信息，请参阅 PUT Object。	否
file	文件或文本内容。 文件或内容必须是表单中的最后一个字段。其下的任何字段都会忽略。 您一次仅能上传一个文件。	是

策略构建

主题

- 过期
- Conditions
- 条件匹配
- 字符转义

策略是使用 UTF-8 和 Base64 编码的 JSON 文档，它指定了请求必须满足的条件并且用于对内容进行身份验证。根据您设计策略文档的方式，您可以对每次上传、每个用户、所有上传或根据其他能够满足您需要的设计来使用它们。

> **Note**
>
> 尽管策略文档是可选的，我们强烈建议您使用它来创建公开可写的存储桶。

下面是策略文档的示例：

```
{ "expiration": "2007-12-01T12:00:00.000Z",

  "conditions": [

    {"acl": "public-read" },

    {"bucket": "awsexamplebucket1" },

    ["starts-with", "$key", "user/eric/"],

  ]

}
```

策略文档包括过期和条件。

过期

过期元素采用 ISO 8601 UTC 日期格式来指定策略的过期日期。例如，“2007-12-01T12:00:00.000Z”指定策略在 2007 年 12 月 1 日午夜 UTC 之后失效。在策略中过期是必需的。

Conditions

策略文档中的条件验证上传的对象的内容。您在表单中指定的每个表单字段 (AWSAccessKeyId、签名、文件、策略和带 x-ignore- 前缀的字段名称除外) 必须包含在条件列表中。

> **Note**
>
> 如果您拥有多个具有相同名称的字段，必须使用逗号分隔值。例如，如果您拥有两个名为“x-amz-meta-tag”的字段，第一个字段的值为“Ninja”，第二个字段的值为“Stallman”，您可以将策略文档设置为 `Ninja,Stallman`。
> 表单中的所有变量会在验证策略之前进行扩展。因此，应该针对扩展的字段执行所有条件匹配。例如，如果您将键字段设置为 `user/betty/${filename}`，您的策略可能是 `[ "starts-with", "$key", "user/betty/" ]`。请勿输入 `[ "starts-with", "$key", "user/betty/${filename}" ]`。有关更多信息，请参阅 条件匹配。

下表介绍策略文档条件。

元素名称	说明
acl	指定 ACL 必须满足的条件。 支持精确匹配和 `starts-with` 。
content-length-range	指定已上传内容允许的最小和最大大小。 支持范围匹配。
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	特定于 REST 的标头。 支持精确匹配和 `starts-with` 。
key	已上传的键的名称。 支持精确匹配和 `starts-with` 。
success_action_redirect, 重定向	上传成功后客户端重定向到的 URL。 支持精确匹配和 `starts-with` 。
success_action_status	

元素名称	说明
	如果没有指定 success_action_redirect，上传成功后状态代码将返回到客户端。 支持精确匹配。
x-amz-security-token	Amazon DevPay 安全令牌。 使用 Amazon DevPay 的每个请求都需要两个 `x-amz-security-token` 表单字段：一个用于产品令牌，一个用于用户令牌。因此，必须使用逗号来分隔值。例如，如果用户令牌是 `eW91dHViZQ==` ，产品令牌是 `b0hnNVNKWVJIQTA=` ，您可以将策略条目设置为：`{ "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" }` 。
其他以 x-amz-meta- 为前缀的字段名称	特定于用户的元数据。 支持精确匹配和 `starts-with` 。

> **Note**
>
> 如果您的工具包添加了其他字段 (例如，Flash 添加了文件名)，您必须将它们添加到策略文档。如果您可以控制此功能，将 `x-ignore-` 添加为字段的前缀以使 Amazon S3 忽略此功能并使其不影响此功能的未来版本。

条件匹配

下表介绍条件匹配类型。尽管您必须为您在表单中指定的每个表单字段指定一个条件，您也可以通过为某个表单字段指定多个条件来创建更复杂的匹配条件。

Condition	说明
精确匹配	精确匹配将验证字段是否匹配特定的值。此示例指示 ACL 必须设置为公共读取：

Condition	说明
	`{"acl": "public-read" }` 此示例是指示 ACL 必须设置为公共读取的替代方法： `[ "eq", "$acl", "public-read" ]`
Starts With	如果值必须从某个特定的值开始，请使用 starts-with。本示例指示键必须以 user/betty 开头： `["starts-with", "$key", "user/betty/"]`
匹配任何内容	要配置策略以允许字段中的任何内容，请使用 starts-with 和一个空值。本示例允许任何 success_action_redirect： `["starts-with", "$success_action_redirect", ""]`
指定范围	对于接受范围的字段，请使用逗号来分隔上限和下限值。本示例允许 1 到 10 MB 的文件大小： `["content-length-range", 1048579, 10485760]`

字符转义

下表介绍策略文档中必须进行转义的字符。

转义序列	说明
\\	反斜杠

转义序列	说明
\$	美元符号
\b	退格键
\f	换页
\n	新建行
\r	回车
\t	水平选项卡
\v	垂直选项卡
\u*xxxx*	所有 Unicode 字符

构建签名

步骤	说明
1	使用 UTF-8 对策略进行编码。
2	使用 Base64 对这些 UTF-8 字节进行编码。
3	使用 HMAC SHA-1，通过您的秘密访问密钥签署策略。
4	使用 Base64 对 SHA-1 签名进行编码。

有关身份验证的一般信息，请参阅Amazon S3 的身份和访问管理。

重定向

本节描述了如何处理重定向。

一般重定向

完成 POST 请求后，用户将重定向至您在 success_action_redirect 字段中指定的位置。如果 Amazon S3 无法解释 URL，它将忽略 success_action_redirect 字段。

如果未指定 success_action_redirect，Amazon S3 将返回在 success_action_status 字段中指定的空文档类型。

如果 POST 请求失败，Amazon S3 将显示错误并且不会提供重定向。

预先上传重定向

如果您的存储桶是使用 <CreateBucketConfiguration> 创建的，您的最终用户可能会需要重定向。如果出现这种要求，某些浏览器可能无法正确处理重定向。这是比较罕见的，但在创建存储桶之后最有可能发生这种情况。

上传示例 (AWS Signature Version 2)

主题

- 文件上传
- 文本区域上传

> **Note**
>
> 本节中讨论的请求身份验证基于 AWS Signature Version 2，这是一种对 AWS 服务的入站 API 请求进行身份验证的协议。
> Amazon S3 现在在所有 AWS 区域 支持 Signature Version 4，后者是一种用于对 AWS 服务入站 API 请求进行身份验证的协议。目前，于 2014 年 1 月 30 日前创建的 AWS 区域 将继续支持之前的协议：Signature Version 2。于 2014 年 1 月 30 日后创建的所有新区域将只支持 Signature Version 4，因此，发往这些区域的所有请求都必须采用 Signature Version 4。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的示例：使用 HTTP POST 的基于浏览器的上传（使用 AWS Signature Version 4）。

文件上传

此示例演示构造用于上传文件附件的策略和表单的完整过程。

策略和表单构建

以下策略支持针对 awsexamplebucket1 存储桶上传到 Amazon S3。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

此策略要求以下内容：

- 上传必须在 2007 年 12 月 1 日 12:00 UTC 之前进行。
- 内容必须上传到 awsexamplebucket1 存储桶。
- 键必须以“user/eric/”开头。
- ACL 必须设置为公共读取。
- success_action_redirect 设置为 https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html。
- 对象是一个图像文件。
- x-amz-meta-uuid 标签必须设置为 14365123651274。
- x-amz-meta-tag 可以包含任意值。

下面是此策略的 Base64 编码版本。

```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidW
```

使用您的凭证创建签名，例如 0RavWzkygo6QX9caELEqKi9kDbU= 是先前的策略文档的签名。

以下格式支持对使用此策略的 amzn-s3-demo-bucket 存储桶发出 POST 请求。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
  ...
  <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
 enctype="multipart/form-data">
    Key to upload: <input type="input" name="key" value="user/eric/" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html" />
    Content-Type: <input type="input" name="Content-Type" value="image/jpeg" /><br />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    File: <input type="file" name="file" /> <br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

示例请求

此请求假设上传的图像为 117,108 字节；不包括图像数据。

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
 Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
```

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: 118698

--9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
--9431149156168
Content-Disposition: form-data; name="acl"

public-read
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html
--9431149156168
Content-Disposition: form-data; name="Content-Type"

image/jpeg
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some,Tag,For,Picture
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--9431149156168
Content-Disposition: form-data; name="Policy"

eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
--9431149156168
Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
```

```
Content-Type: image/jpeg

...file content...
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

示例响应

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html?bucket=awsexamplebucket1&key=user/eric/
MyPicture.jpg&etag=&quot;39d459dfbc0faabbb5e179358dfb94c3&quot;
Server: AmazonS3
```

文本区域上传

主题

- 策略和表单构建
- 示例请求
- 示例响应

以下示例演示构造策略和表单以上传文本区域的完整过程。上传文本区域对于提交用户创建的内容 (如博客文章) 十分有用。

策略和表单构建

以下策略支持针对 awsexamplebucket1 存储桶将文本区域上传到 Amazon S3。

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
```

```
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

此策略要求以下内容：

- 上传必须在 2007 年 12 月 1 日 12:00 GMT 之前进行。
- 内容必须上传到 awsexamplebucket1 存储桶。
- 键必须以“user/eric/”开头。
- ACL 必须设置为公共读取。
- success_action_redirect 设置为 https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html。
- 对象是 HTML 文本。
- x-amz-meta-uuid 标签必须设置为 14365123651274。
- x-amz-meta-tag 可以包含任意值。

下面是此策略的 Base64 编码版本。

```
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXR
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXRoIn0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCAidXNlci
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2p
C5zMy5hbWF6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSIsICJ0ZXh0L2h0
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFtei1t
IsICIiXQogIF0KfQo=
```

使用您的凭证创建签名。例如，qA7FWXKq6VvU68lI9KdveT1cWgF= 是先前的策略文档的签名。

以下格式支持对使用此策略的 amzn-s3-demo-bucket 存储桶发出 POST 请求。

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
```

```
  <body>
  ...
  <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
 enctype="multipart/form-data">
    Key to upload: <input type="input" name="key" value="user/eric/" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html" />
    <input type="hidden" name="Content-Type" value="text/html" />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
    <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
    <input type="hidden" name="Policy" value="POLICY" />
    <input type="hidden" name="Signature" value="SIGNATURE" />
    Entry: <textarea name="file" cols="60" rows="10">

Your blog post goes here.

    </textarea><br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

示例请求

此请求假设上传的图像为 117,108 字节；不包括图像数据。

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
 Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=178521717625888
Content-Length: 118635

-178521717625888
```

```
Content-Disposition: form-data; name="key"

ser/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html
--178521717625888
Content-Disposition: form-data; name="Content-Type"

text/html
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
--178521717625888
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--178521717625888
Content-Disposition: form-data; name="Policy"
eyAiZXhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN

--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
```

```
--178521717625888--
```

示例响应

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket1&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

POST 和 Adobe Flash

本节描述了如何使用 POST 和 Adobe Flash。

Adobe Flash Player 的安全性

在默认情况下，Adobe Flash Player 安全模型禁止 Adobe Flash Players 创建指向位于域 (服务 SWF 文件) 之外的服务器的网络连接。

要覆盖默认设置，您必须将公共可读的 crossdomain.xml 文件上传到接受 POST 上传的存储桶。下面是示例 crossdomain.xml 文件。

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

> **Note**
>
> 有关 Adobe Flash 安全模型的更多信息，请转到 Adobe 网站。
> 向存储桶添加 crossdomain.xml 文件可允许所有 Adobe Flash Player 连接到您存储桶中的 crossdomain.xml 文件；但是，它不会授予对实际 Amazon S3 存储桶的访问权限。

Adobe Flash 注意事项

Adobe Flash 中的 FileReference API 将 `Filename` 表单字段添加到 POST 请求。在构建使用 FileReference API 操作上传到 Amazon S3 的 Adobe Flash 应用程序时，请在策略中包括下面的条件：

```
['starts-with', '$Filename', '']
```

某些版本的 Adobe Flash Player 无法正确处理带空白正文的 HTTP 响应。要将 POST 配置为返回不带空白正文的响应，请将 `success_action_status` 设置为 201。Amazon S3 随后将返回一个具有 201 状态代码的 XML 文档。有关 XML 文档内容的信息，请参阅 POST Object。有关表单字段的信息，请参阅 HTML 表单字段。

最佳实践设计模式：优化 Amazon S3 性能

当从 Amazon S3 上传和检索存储时，您的应用程序可以轻松地实现每秒数千个事务的请求性能。Amazon S3 自动扩展到高请求速率。例如，您的应用程序可以实现每秒每个分区的 Amazon S3 前缀至少 3500 个PUT/COPY/POST/DELETE 或 5500 个 GET/HEAD 请求。对存储桶中的前缀数量没有限制。您可以通过使用并行来增加读取或写入性能。例如，如果您在 Amazon S3 存储桶中创建 10 个前缀以并行处理读取，则可以将读取性能扩展到每秒 55,000 个读取请求。同样，您可以通过写入多个前缀来扩展写入操作。无论是读取操作还是写入操作，扩展都是逐渐发生的，而不是瞬间发生的。当 Amazon S3 扩展到新的更高请求速率时，您可能会看到一些 503（减速）错误。扩展完成后，这些错误将消失。有关创建和使用前缀的更多信息，请参阅使用前缀组织对象。

Amazon S3 上的某些数据湖应用程序对于运行超过 PB 级数据的查询扫描数百万或数十亿个对象。这些数据湖应用程序实现的单一实例传输速率可最大限度地提高 Amazon EC2 实例的网络接口利用率，这在单一实例上可高达 100 Gb/s。然后，这些应用程序跨多个实例聚合吞吐量，以获得每秒多个 Tb 的级别。

另外一些应用程序对延迟很敏感，例如社交媒体消息传递应用程序。这些应用程序可实现一致的小对象延迟（对于较大对象，为第一个字节输出延迟），延迟时间大约为 100 - 200 毫秒。

其他 AWS 服务也可帮助加快不同应用程序架构的性能。例如，如果您希望通过单一 HTTP 连接实现较高的传输速率，或需要单一位毫秒延迟，请使用 Amazon CloudFront 或 Amazon ElastiCache 以通过 Amazon S3 进行缓存。

此外，如果您希望在客户端与 S3 存储桶之间获得较快的长距离数据传输速度，请使用 使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输。Transfer Acceleration 使用 CloudFront 中的全球分布式边缘站点来加快跨地理距离的数据传输。如果 Amazon S3 工作负载使用具有 AWS KMS 的服务器端加密，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS 限制，以获取有关使用案例支持的请求速率的信息。

下面的主题介绍的最佳实践准则和设计模式用于优化使用 Amazon S3 的应用程序的性能。有关对 Amazon S3 进行性能优化的最新信息，请参阅 Amazon S3 的性能准则 和 Amazon S3 的性能设计模式。

> **Note**
>
> 有关将 Amazon S3 Express One Zone 存储类与目录存储桶配合使用的更多信息，请参阅 什么是 S3 Express One Zone？ 和目录桶。

主题

Amazon S3 的性能准则

当构建用于从 Amazon S3 上传和检索对象的应用程序时，请遵循我们的最佳实践准则以优化性能。我们还提供了更详细的 性能设计模式。

要在 Amazon S3 上获得应用程序的最佳性能，我们建议遵循以下准则。

主题

衡量性能

当优化性能时，查看网络吞吐量、CPU 和 DRAM 要求。根据针对这些不同资源的组合需求，可能需要评估不同的 Amazon EC2 实例类型。有关实例类型的信息，请参阅《Amazon EC2 用户指南》中的实例类型。

它还有助于在衡量性能时使用 HTTP 分析工具查看 DNS 查找时间、延迟和数据传输速度。

要了解性能要求并优化应用程序的性能，还可以监控收到的 503 错误响应。监控某些性能指标可能会产生额外费用。有关更多信息，请参阅 Amazon S3 定价。

监控 503（减速）状态错误响应的数量

要监控收到的 503 状态错误响应的数量，可以使用以下选项之一：

- 使用 Amazon S3 的 Amazon CloudWatch 请求指标。CloudWatch 请求指标包括一个用于 5xx 状态响应的指标。有关 CloudWatch 请求指标的更多信息，请参阅使用 Amazon CloudWatch 监控指标。
- 使用 Amazon S3 Storage Lens 存储统计管理工具的高级指标部分中提供的 503（服务不可用）错误计数。有关更多信息，请参阅 使用 S3 Storage Lens 存储统计管理工具指标提高性能。
- 使用 Amazon S3 服务器访问日志记录。通过服务器访问日志记录，您可以筛选和查看所有收到 503（内部错误）响应的请求。还可以使用 Amazon Athena 解析日志。有关服务器访问日志记录的更多信息，请参阅 使用服务器访问日志记录来记录请求。

通过监控 HTTP 503 状态错误代码的数量，您通常可以获得宝贵的见解，了解哪些前缀、密钥或存储桶获得的节流请求最多。

横向扩展存储连接

跨许多连接分布请求是一种常用的横向扩展性能的设计模式。当构建高性能应用程序时，将 Amazon S3 视为非常大的分布式系统，而不是类似于传统存储服务器的单个网络端点。您可以通过将多个并行请求发到 Amazon S3 来实现最佳性能。将这些请求分布在不同的连接，以最大限度地利用 Amazon S3 的可访问带宽。Amazon S3 对与存储桶建立的连接数没有任何限制。

使用字节范围提取

通过在 GET Object 请求中使用 Range HTTP 标头，您可以从对象中提取字节范围，而只传输指定的部分。您可以使用到 Amazon S3 的并行连接，从相同对象中提取不同的字节范围。这有助于您通过单一整个对象请求实现更高的聚合吞吐量。通过提取较小范围的大型对象，您的应用程序还可以在请求中断时改善重试次数。有关更多信息，请参阅 下载对象。

字节范围请求的典型大小为 8 MB 或 16 MB。如果使用分段上传来 PUT 对象，则最佳实践是以相同的分段大小来 GET 它们（或者至少与分段边界相符）以获得最佳性能。GET 请求可以直接面向单独的分段；例如，`GET ?partNumber=N`.

延迟敏感型应用程序的重试请求

大量的超时和重试会导致持续不断的延迟。考虑到 Amazon S3 的大规模，如果第一个请求较慢，则退出的请求可能采取不同的路径并快速成功。AWS SDK 具有可配置的超时和重试值，您可以进行调整以符合特定应用程序的容限。

在同一 AWS 区域 中结合 Amazon S3（存储）和 Amazon EC2（计算）

尽管 S3 存储桶名称是全局唯一的，但每个存储桶都存储在您创建它时所选择的区域中。为了优化性能，我们建议您尽可能从位于同一 AWS 区域的 Amazon EC2 实例中访问此存储桶。这可以减少网络延迟和数据传输成本。

有关数据传输成本的更多信息，请参阅 Amazon S3 定价。

使用 Amazon S3 Transfer Acceleration 最大限度地减少因距离导致的延迟

使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输 可在客户端与 S3 存储桶之间管理快速、轻松和安全的长地理距离文件传输。Transfer Acceleration 利用 Amazon CloudFront 中的全球分布式边缘站点。当数据到达某个边缘站点时，会通过经过优化的网络路径路由至 Amazon S3。Transfer Acceleration 适合定期跨大洲传输 GB 到 TB 的数据。它也适用于全球各地需要上传到集中式存储桶的客户。

您可以使用 Amazon S3 Transfer Acceleration 速度比较工具来比较各个 Amazon S3 区域内加快的上传速度和未加快的上传速度。此速度比较工具使用分段上传来将文件从浏览器传输到各种使用和未使用 Amazon S3 Transfer Acceleration 的 Amazon S3 区域。

使用最新版本的 AWS SDK

AWS SDK 为许多用于优化 Amazon S3 性能的建议准则提供内置的支持。这些 SDK 提供了更简单的 API 以便从应用程序内部利用 Amazon S3，并定期更新以遵循最新的最佳实践。例如，这些 SDK 包含自动对 HTTP 503 错误重试请求的逻辑，并投资编写代码来应对和适应慢速连接。

这些 SDK 还提供传输管理器，以便在适当的时机使用字节范围请求来自动横向扩展连接，从而实现每秒数千个请求。务必使用最新版本的 AWS SDK，以获取最新的性能优化特征。

您还可以在使用 HTTP REST API 请求时优化性能。当使用 REST API 时，您应遵循属于 SDK 一部分的相同最佳实践。对于慢速连接允许超时和重试，并使用多个连接以允许并行提取对象数据。有关使用 REST API 的信息，请参阅 Amazon Simple Storage Service API 参考。

Amazon S3 的性能设计模式

当设计应用程序以从 Amazon S3 上传和检索对象时，使用我们的最佳实践设计模式，以实现应用程序的最佳性能。我们还提供 性能准则，供您在规划您的应用程序架构时考虑。

要优化性能，您可以使用以下设计模式。

主题

对频繁访问的内容使用缓存

许多在 Amazon S3 中存储数据的应用程序会提供用户反复请求的数据的一个“工作集”。如果工作负载针对一组常用对象发送重复的 GET 请求，则可以使用 Amazon CloudFront、Amazon ElastiCache 或 AWS Elemental MediaStore 等缓存来优化性能。成功地采用缓存可能导致低延迟和较高的数据传输速率。使用缓存的应用程序还会向 Amazon S3 发送少量的直接请求，这可帮助降低请求成本。

Amazon CloudFront 是一个快速内容交付网络 (CDN)，它透明地在一大组地理位置分散的节点 (PoP) 中缓存 Amazon S3 中的数据。当可能从多个区域或通过 Internet 访问对象时，CloudFront 允许在访问对象的用户附近缓存数据。这样就可以实现常见 Amazon S3 内容的高性能交付。有关 CloudFront 的更多信息，请参阅 Amazon CloudFront 开发人员指南。

Amazon ElastiCache 是一个托管的内存中缓存。通过 ElastiCache，您可以预配置在内存中缓存对象的 Amazon EC2 实例。这种缓存会导致 GET 延迟下降若干个数量级，并显著增加下载吞吐量。要使用 ElastiCache，请修改应用程序逻辑，以便用热门对象填充缓存和检查缓存中的热门对象，然后从 Amazon S3 中请求它们。有关使用 ElastiCache 提高 Amazon S3 GET 性能的示例，请参阅博客文章使用 Amazon ElastiCache for Redis 增强 Amazon S3。

AWS Elemental MediaStore 是一个专为 Amazon S3 中的视频工作流和媒体交付构建的缓存和内容分配系统。MediaStore 提供专用于视频的端到端存储 API，推荐用于性能敏感型视频工作负载。有关 MediaStore 的更多信息，请参阅 AWS Elemental MediaStore 用户指南。

延迟敏感型应用程序的超时和重试

在某些情况下，应用程序会收到来自 Amazon S3 的响应，指示需要重试。Amazon S3 将存储桶和对象名称映射到与其关联的对象数据。如果应用程序生成高的请求率（通常针对少量对象持续超过每秒 5,000 个请求的速率），则它可能会收到 HTTP 503 速度下降 响应。如果出现这些错误，每个 AWS SDK 都会使用指数退避来实现自动重试逻辑。如果您不使用 AWS SDK，则应在收到 HTTP 503 错误时实施重试逻辑。有关回退技术的信息，请参阅《Amazon Web Services 一般参考》中的 AWS 中的错误重试和指数回退。

Amazon S3 自动扩展以应对持续的新请求速率，同时动态地优化性能。尽管 Amazon S3 在内部针对新的请求速率进行优化，但您将临时收到 HTTP 503 请求响应，直至优化完成。当 Amazon S3 在内部针对新的请求速率优化性能后，通常无需重试就能正常应对所有请求。

对于延迟密集型应用程序，Amazon S3 建议跟踪和主动重试较慢的操作。当重试请求时，我们建议您使用到 Amazon S3 的新连接并执行全新 DNS 查找。

当您发出大型可变大小的请求（例如，超过 128 MB）时，我们建议跟踪所实现的吞吐量并重试最慢 5% 的请求。当您发出较小的请求（例如，小于 512 KB）时，其中，延迟时间中值通常处于数十毫秒的范围内，好的指导原则是在 2 秒后重试 GET 或 PUT 操作。如果需要更多重试，则最佳实践是退避。例如，我们建议您在 2 秒后发出一次重试，然后 4 秒后再发出一次重试。

如果应用程序对 Amazon S3 发出固定大小的请求，则应预期对于其中每个请求获得更一致的响应时间。在这种情况下，一个简单的策略是确定最慢的 1% 的请求并重试这些请求。甚至一次重试也会频繁、有效地减少延迟。

如果您使用 AWS Key Management Service (AWS KMS) 进行服务器端加密，请参阅 AWS Key Management Service 开发人员指南中的限制，以获取有关使用案例支持的请求速率的信息。

横向扩展和请求并行化以实现高吞吐量

Amazon S3 是一个非常大的分布式系统。为了帮助您利用其规模，我们建议您将并行请求横向扩展到 Amazon S3 服务端点。除了在 Amazon S3 中分布请求之外，这种类型的扩展方法还有助于将负载分布到整个网络中的多个路径。

对于高吞吐量传输，Amazon S3 建议使用并行利用多个连接来 GET 或 PUT 数据的应用程序。例如，AWS Java SDK 中的 Amazon S3 Transfer Manager 支持上述功能，大多数其他 AWS SDK 也提供类似的构造。对于某些应用程序，您可以实现并行连接，具体方法为：在不同的应用程序线程或不同的应用程序实例中并行启动多个连接。要采取的最佳方法取决于应用程序以及您访问的对象的结构。

您可以使用 AWS SDK 直接发出 GET 和 PUT 请求，而不是利用 AWS SDK 中的传输管理。这种方法让您可以更直接地调整工作负载，同时仍受益于软件工具包支持重试和处理任何可能发生的 HTTP 503 响应。通常来说，当您将某个区域中的大型对象从 Amazon S3 下载到 Amazon EC2 时，我们建议您以 8-16 MB 的粒度向对象的字节范围发出并行请求。对于每个 85-90 MB/s 的所需网络吞吐量发出一个并行请求。要使 10 Gb/s 网络接口卡 (NIC) 达到饱和，您可以通过单独的连接使用大约 15 个并行请求。您可以通过更多连接扩展并行请求以使速度更快的 NIC 达到饱和，如 25 Gb/s 或 100 Gb/s NIC。

当您调整并行发出的请求数量时，衡量性能至关重要。我们建议一次只用一个请求为起点。衡量所实现的网络带宽以及应用程序用于处理数据的其他资源的使用情况。然后，您可以确定瓶颈资源（也即使用率最高的资源），进而确定可能有用的请求数。例如，如果一次处理一个请求会导致 CPU 利用率达到

25%，则建议可以容纳多达四个并行请求。衡量是必不可少的，随着请求速率增加，务必确认资源使用情况。

如果应用程序使用 REST API 直接向 Amazon S3 发出请求，我们建议您使用 HTTP 连接池，并对一系列请求重用每个连接。通过避免逐个请求连接设置，可消除对每个连接执行 TCP 慢速启动和安全套接字层 (SSL) 握手的需要。有关使用 REST API 的信息，请参阅 Amazon Simple Storage Service API 参考。

最后，务必注意 DNS 并仔细检查请求正在分散到广泛的 Amazon S3 IP 地址池。对于 Amazon S3 的 DNS 查询会循环经过一个很大的 IP 端点列表。但是，缓存解析程序或重用单一 IP 地址的应用程序代码不会受益于地址多样性和由此产生的负载平衡。网络实用工具（如 `netstat` 命令行工具）可以显示用来与 Amazon S3 通信的 IP 地址，并且我们提供要使用的 DNS 配置的准则。有关这些准则的更多信息，请参阅提出请求。

使用 Amazon S3 Transfer Acceleration 加快地理位置分散的数据传输

使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输 通过使用 Amazon S3，有效地将由于全球分散的客户端与区域性应用程序之间的地理距离所导致的延迟降至最小甚至消除。Transfer Acceleration 使用 CloudFront 中的全球分布式边缘站点进行数据传输。AWS 边缘网络在超过 50 个站点具有节点。目前，它用于通过 CloudFront 分布内容，并向针对 Amazon Route 53 发出的 DNS 查询提供快速响应。

边缘网络还有助于加速进出 Amazon S3 的数据传输。它对具有以下特点的应用程序是理想之选：跨大洲或在大洲之间传输数据、具有快速 Internet 连接、使用大型对象，或具有大量要上传的内容。当数据到达某个边缘站点时，数据会被经过优化的网络路径路由至 Amazon S3。通常，您与 Amazon S3 区域之间的距离越远，使用 Transfer Acceleration 可预期获得的速度改进就越大。

您可以对新的或现有的存储桶设置 Transfer Acceleration。您可以通过单独的 Amazon S3 Transfer Acceleration 端点来使用 AWS 边缘站点。测试 Transfer Acceleration 是否有助于提高客户端请求性能的最佳方法是使用 Amazon S3 Transfer Acceleration 速度比较工具。网络配置和条件随时间或站点不同而可能发生变化。因此，我们只对 Amazon S3 Transfer Acceleration 可能会改进上传性能的传输向您收费。有关将 Transfer Acceleration 与不同的 AWS SDK 结合使用的信息，请参阅 启用和使用 S3 Transfer Acceleration。

什么是 Amazon S3 on Outposts？

AWS Outposts 是一项完全托管式服务，它为几乎任何数据中心、主机托管空间或本地部署设施提供相同的 AWS 基础架构、AWS 服务、API 和工具，以实现真正一致的混合体验。AWS Outposts 非常适合需要低延迟访问本地部署系统、本地数据处理、数据驻留以及迁移具有本地系统相互依赖项的应用程序的工作负载。有关更多信息，请参阅AWS Outposts《用户指南》中的什么是AWS Outposts？。

借助 Amazon S3 on Outposts，您可以在 Outposts 上创建 S3 桶并在本地中轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 OUTPOSTS，该存储类使用 Amazon S3 API，并且用于在 Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过虚拟私有云（VPC）使用访问点和端点连接与 Outposts 桶进行通信。

您可以像在 Amazon S3 中一样在 Outposts 存储桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。

- S3 on Outposts 的工作原理
- S3 on Outposts 的功能
- 相关服务
- 访问 S3 on Outposts
- 支付 S3 on Outposts 的费用
- 后续步骤

S3 on Outposts 的工作原理

S3 on Outposts 是一种对象存储服务，可将数据以对象形式存储在 Outpost 上的存储桶中。对象 是指一个数据文件和描述该文件的任何元数据。存储桶是对象的容器。

要将数据存储在 S3 on Outposts 中，您需要先创建存储桶。创建存储桶时，您可以指定存储桶名称和将容纳该存储桶的 Outpost。要访问 S3 on Outposts 存储桶并执行对象操作，接下来要创建并配置访问点。您还必须创建端点以将请求路由到访问点。

访问点简化了在 S3 中存储数据的任何 AWS 服务或客户应用程序的数据访问。访问点是附加到存储桶的命名网络端点，您可以使用这些存储桶执行对象操作（如 `GetObject` 和 `PutObject`）。每个访问点都有不同的权限和网络控制。

您可以使用 AWS Management Console、AWS CLI、AWS SDK 或 REST API 创建和管理 S3 on Outposts 存储桶、访问点和端点。要上传和管理 S3 on Outposts 存储桶中的对象，您可以使用AWS CLI、AWS SDK 或 REST API。

区域

在 AWS Outposts 调配期间，您或 AWS 将创建服务链路连接，将您的 Outpost 连接回您选择的 AWS 区域 或者用于存储桶操作和遥测的 Outposts 所属区域。Outposts 依赖于与父 AWS 区域 的连接。Outposts 机架不是为断开连接的操作或连接有限或没有连接的环境而设计的。有关更多信息，请参阅《AWS Outposts 用户指南》中的与 AWS 区域 的 Outpost 连接。

存储桶

存储桶是用于 S3 on Outposts 中存储的对象的容器。您可以在存储桶中存储任意数量的对象，并且每个 Outpost 每个账户中最多可以有 100 个存储桶。

创建存储桶时，您可以输入存储桶名称，然后选择存储桶将驻留的 Outpost。创建存储桶后，无法更改存储桶名称或将存储桶移到不同的 Outpost。存储桶名称必须遵循 Amazon S3 存储桶命名规则。在 S3 on Outposts 中，存储桶名称对于 Outposts 和 AWS 账户 是唯一的。S3 on Outposts 存储桶需要 `outpost-id`、`account-id` 以及用于识别它们的存储桶名称。

以下示例显示了用于 S3 on Outposts 存储桶的 Amazon Resource Name（ARN）格式。ARN 包括 Outpost 所属的区域、您的 Outpost 账户、Outpost ID 和存储桶名称。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

每个对象都储存在一个存储桶中。您必须使用访问点访问 Outposts 存储桶中的任何对象。当您为对象操作指定桶时，可以使用访问点 ARN 或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

以下示例显示了 S3 on Outposts 的访问点 ARN 格式，其中包括 `outpost-id`、`account-id` 和访问点名称：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

有关存储桶的更多信息，请参阅 使用 S3 on Outposts 存储桶。

对象

对象是 S3 on Outposts 中存储的基础实体。对象由对象数据和元数据组成。元数据是一组描述对象的名称-值对。这些对值包括一些默认元数据（如上次修改日期）和标准 HTTP 元数据（如 `Content-Type`）。您还可以在存储对象时指定自定义元数据。存储桶中的对象将由键（或名称）进行唯一地标识。

对于 Amazon S3 on Outposts，对象数据始终存储在 Outpost 上。当 AWS 安装 Outpost 机架时，您的数据将保留在 Outpost 的本地，以满足数据驻留要求。您的对象永远不会离开您的 Outpost，也不在 AWS 区域 中。由于 AWS Management Console 托管在区域内，您无法使用控制台上传或管理 Outpost 中的对象。但是，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 和 AWS SDK 通过访问点上传和管理对象。

键

对象密钥（或密钥名称）是指存储桶中对象的唯一标识符。存储桶内的每个对象都只能有一个键。存储桶和对象键的组合唯一标识各个对象。

以下示例显示了 S3 on Outposts 对象的 ARN 格式，其中包括 Outpost 所在区域的 AWS 区域 代码、AWS 账户 ID、Outpost ID、存储桶名称和对象键：

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket1/object/myobject
```

有关对象键的更多信息，请参阅 使用 S3 on Outposts 对象。

S3 版本控制

您可以对 Outposts 桶使用 S3 版本控制功能，以将对象的多个变体保留在同一桶中。使用 S3 版本控制功能，您可以保留、检索和恢复存储桶中的各个版本。S3 版本控制功能可帮助您从用户意外操作和应用程序故障中恢复。

有关更多信息，请参阅 为 S3 on Outposts 桶管理 S3 版本控制。

版本 ID

在桶中启用 S3 版本控制时，S3 on Outposts 会为添加到桶中的每个对象生成唯一的版本 ID。启用版本控制时存在于存储桶中的对象的版本 ID 为`null`。如果使用其他操作修改这些（或任何其他）对象，例如 PutObject，则新对象将获得唯一的版本 ID。

有关更多信息，请参阅 为 S3 on Outposts 桶管理 S3 版本控制。

存储类别和加密

S3 on Outposts 提供了一个新的存储类 S3 Outposts (OUTPOSTS)。S3 Outposts 存储类仅可用于存储在 AWS Outposts 上存储桶中的对象。如果您尝试将其他 S3 存储类与 S3 on Outposts 一起使用，则 S3 on Outposts 将返回 `InvalidStorageClass` 错误消息。

默认情况下，存储在 S3 Outposts (OUTPOSTS) 存储类中的对象使用服务器端加密方式和由 Amazon S3 托管的加密密钥 (SSE-S3) 进行加密。有关更多信息，请参阅 S3 on Outposts 中的数据加密。

存储桶策略

存储桶策略是基于资源的 AWS Identity and Access Management (IAM) 策略，您可以使用该策略向存储桶及其中对象授予访问权限。只有存储桶拥有者才能将策略与存储桶关联。附加到存储桶的权限适用于存储桶拥有者拥有的存储桶中所有对象。存储桶策略的大小限制为 20 KB。

存储桶策略使用基于 JSON 的 IAM 策略语言，该语言是跨 AWS 的标准语言。您可以使用存储桶策略添加或拒绝存储桶中对象的权限。存储桶策略基于策略中的元素允许或拒绝请求。这些元素可以包括请求者、S3 on Outposts 操作、资源以及请求的特征或条件（例如，用于发出请求的 IP 地址）。例如，您可以创建一个存储桶策略，该策略授予跨账户将对象上传到 S3 on Outposts 存储桶的权限，同时确保存储桶拥有者对上传的对象拥有完全控制权。有关更多信息，请参阅 Amazon S3 存储桶策略的示例。

在存储桶策略中，您可以在 ARN 和其他值上使用通配符 (*) 来授予对对象子集的权限。例如，您可以控制对以通用前缀或以给定扩展名结尾的对象组的访问，例如 `.html`。

S3 on Outposts 访问点

S3 on Outposts 访问点被命名为网络端点，其专用访问策略描述了如何使用该端点访问数据。访问点可简化对 S3 on Outposts 中的共享数据集的大规模数据访问管理。访问点附加到存储桶，您可以使用这些存储桶执行 S3 对象操作（如 `GetObject` 和 `PutObject`）。

当您为对象操作指定桶时，可以使用访问点 ARN 或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

访问点具有不同的权限和网络控制，S3 on Outposts 将它们应用于通过该访问点发出的任何请求。每个接入点强制实施自定义接入点策略，该策略与附加到底层存储桶的存储桶策略结合使用。

有关更多信息，请参阅 访问 S3 on Outposts 存储桶和对象。

S3 on Outposts 的功能

访问管理

S3 on Outposts 提供了用于审核和管理对存储桶和对象的访问的功能。默认情况下，S3 on Outposts 存储桶和对象都是私有的。您只能访问您创建的 S3 on Outposts 资源。

要授予支持您的特定使用案例的细粒度资源权限或审核 S3 on Outposts 资源的权限，您可以使用以下功能。

- S3 阻止公有访问 - 阻止对存储桶和对象的公有访问。对于 Outposts 上的存储桶，默认情况下始终启用 Block Public Access（阻止公共访问）。
- AWS Identity and Access Management（IAM） – IAM 是一种 Web 服务，可帮助您安全地控制对 AWS 资源（包括 S3 on Outposts 资源）的访问。借助 IAM，您可以集中管理控制用户可访问哪些 AWS 资源的权限。可以使用 IAM 来控制谁通过了身份验证（准许登录）并获得授权（具有相应权限）来使用资源。
- S3 on Outposts 访问点 – 管理对 S3 on Outposts 中的共享数据集的数据访问。访问点是具有专用访问策略的命名网络端点。访问点附加到存储桶，可用于执行对象操作（如 `GetObject` 和 `PutObject`）。
- 存储桶策略 - 使用基于 IAM 的策略语言为 S3 存储桶及其中的对象配置基于资源的权限。
- AWS Resource Access Manager (AWS RAM) – 跨您的组织或 AWS Organizations 中组织单位 (OU) 内的 AWS 账户 安全共享 S3 on Outposts 容量。

存储日志记录和监控

S3 on Outposts 提供日志记录和监控工具，您可以使用这些工具来监控和控制 S3 on Outposts 资源的使用情况。更多信息，请参阅监控工具。

- 针对 S3 on Outposts 的 Amazon CloudWatch 指标 – 跟踪资源的运行状况并了解容量可用性。
- 针对 S3 on Outposts 的 Amazon CloudWatch Events 事件 – 为任何 S3 on Outposts API 事件创建规则，以便通过所有受支持的 CloudWatch Events 目标接收通知，包括 Amazon Simple Queue Service (Amazon SQS)、Amazon Simple Notification Service (Amazon SNS) 和 AWS Lambda。
- S3 on Outposts 的 AWS CloudTrail 日志 – 记录用户、角色或 S3 on Outposts 中的 AWS 服务 执行的操作。CloudTrail 日志为您提供了 S3 存储桶级和对象级操作的详细 API 跟踪。

强一致性

S3 on Outposts 为所有 AWS 区域 内的 S3 on Outposts 存储桶中对象的 PUT 和 DELETE 请求提供了强大的先写后读一致性。这个行为既适用于新对象的写入，也适用于覆盖现有对象的 PUT 请求以及 DELETE 请求。此外，S3 on Outposts 对象标签和对象元数据（例如 HEAD 对象）具有严格的一致性。有关更多信息，请参阅 Amazon S3 数据一致性模型。

相关服务

将数据加载到 S3 on Outposts 中之后，您可以将数据用于其他 AWS 服务。以下是您可能最常用使用的服务：

- Amazon Elastic Compute Cloud (Amazon EC2) – 在 AWS Cloud中提供安全可扩展的计算容量。使用 Amazon EC2 可减少前期的硬件投入，因此您能够快速开发和部署应用程序。您可以使用 Amazon EC2 启动所需数量的虚拟服务器，配置安全性和联网以及管理存储。
- Outposts 上的 Amazon Elastic Block Store (Amazon EBS) – 使用 Outposts 上的 Amazon EBS 本地快照将卷快照存储在 S3 on Outposts 本地。
- Outposts 上的 Amazon Relational Database Service (Amazon RDS) – 使用 Amazon RDS 本地备份将您的 Amazon RDS 备份存储在您的 Outpost 本地。
- AWS DataSync – 在 Outposts 和 AWS 区域 之间自动传输数据，同时选择要传输的内容、何时传输以及要使用多少网络带宽。S3 on Outposts 已与 AWS DataSync 集成。对于需要高吞吐量本地处理的本地应用程序，S3 on Outposts 提供了本地对象存储，以最大限度地减少网络变化造成的数据传输和缓冲区，同时让您能够轻松地在 Outposts 和 AWS 区域 之间传输数据。

访问 S3 on Outposts

您可以通过以下任何方式使用 S3 on Outpost：

AWS Management Console

控制台是基于 Web 的用户界面，用于管理 S3 on Outposts 和 AWS 资源。如果您已注册 AWS 账户，可以通过登录 AWS Management Console 并从 AWS Management Console 主页中选择 S3 来访问 S3 on Outposts。然后，从左侧导航窗格中选择 Outposts buckets（Outposts 存储桶）。

AWS Command Line Interface

可以使用 AWS 命令行工具，在系统的命令行中发出命令来执行 AWS （包括 S3）任务。

AWS Command Line Interface (AWS CLI) 针对大量 AWS 服务 提供了相关命令。AWS CLI 在 Windows、macOS 和 Linux 上受支持。要开始使用，请参阅 AWS Command Line Interface 用户指南。有关您可用于 S3 on Outposts 的命令的更多信息，请参阅《AWS CLI 命令参考》中的 s3api、s3control 和 s3outposts。

AWS 软件开发工具包

AWS 提供的 SDK（软件开发工具包）包含各种编程语言和平台（Java、Python、Ruby、.NET、iOS、Android 等）的库和示例代码。AWS SDK 提供便捷的方式来创建对 S3 on Outposts 和 AWS 的编程访问。由于 S3 on Outposts 与 Amazon S3 使用相同的 SDK，因此 S3 on Outposts 使用相同的 S3 API、自动化和工具提供一致的体验。

S3 on Outposts 是一项 REST 服务。您可以使用 AWS SDK 库向 S3 on Outposts 发送请求，这些库包装底层 REST API 并简化编程任务。例如，SDK 负责计算签名、加密签名请求、管理错误和自动重试请求等任务。有关 AWS SDK 的信息（包括如何下载及安装），请参阅在 AWS 上构建所用的工具。

支付 S3 on Outposts 的费用

您可以购买各种 AWS Outposts 机架配置，这些配置具有 Amazon EC2 实例类型、Amazon EBS 通用固态硬盘 (SSD) 卷 (gp2) 和 S3 on Outposts 的组合。定价包括交付、安装、基础设施服务维护以及软件修补程序和升级。

有关更多信息，请参阅 AWS Outposts 机架定价。

后续步骤

有关使用 S3 on Outposts 的更多信息，请参阅以下主题：

- 设置 Outpost
- Amazon S3 on Outposts 与 Amazon S3 有何不同？
- 开始使用 Amazon S3 on Outposts
- S3 on Outposts 的网络
- 使用 S3 on Outposts 存储桶
- 使用 S3 on Outposts 对象
- S3 on Outposts 中的安全性

- 管理 S3 on Outposts 存储
- 使用 Amazon S3 on Outposts 进行开发

设置 Outpost

要开始使用 Amazon S3 on Outposts，您需要在您的设施中部署具有 Amazon S3 容量的 Outpost。有关订购 Outpost 和 S3 容量的选项的信息，请参阅 AWS Outposts。要检查 Outposts 在其上是否具有 S3 容量，可以使用 ListOutpostsWithS3 API 调用。有关规格以及要了解 S3 on Outposts 与 Amazon S3 的不同之处，请参阅 Amazon S3 on Outposts 与 Amazon S3 有何不同？

有关更多信息，请参阅以下主题。

主题

- 订购新 Outpost

订购新 Outpost

如果您需要订购具有 S3 容量的新 Outpost，请参阅 AWS Outposts 机架定价以了解适用于 Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Elastic Block Store (Amazon EBS) 和 Amazon S3 的容量选项。

选择配置后，按照《AWS Outposts 用户指南》中创建 Outpost 和订购 Outpost 容量中的步骤操作。

Amazon S3 on Outposts 与 Amazon S3 有何不同？

Amazon S3 on Outposts 向您的本地 AWS Outposts 环境提供对象存储。使用 S3 on Outposts，可通过将数据保持在本地应用程序附近，帮助您满足本地处理、数据驻留和苛刻的性能需求。由于使用 Amazon S3 API 和功能，S3 on Outposts 可以轻松存储、保护、标记、报告和控制对 Outposts 数据的访问，并将 AWS 基础设施扩展到您的本地设施，以获得一致的混合体验。

有关 S3 on Outposts 的独特之处的更多信息，请参阅以下主题。

主题

- S3 on Outposts 规范
- S3 on Outposts 支持的 API 操作
- S3 on Outposts 不支持的 Amazon S3 功能

- S3 on Outposts 网络要求

S3 on Outposts 规范

- Outposts 存储桶的最大大小为 50 TB。
- 每个 AWS 账户 的最大 Outpost 存储桶数为 100。
- Outpost 存储桶只能使用访问点和终端节点进行访问。
- 每个 Outposts 存储桶的最大访问点数为 10。
- 访问点策略的大小限制为 20 KB。
- Outpost 拥有者可以使用 AWS Resource Access Manager 在 AWS Organizations 中管理您组织内的访问权限。需要访问 Outpost 的所有账户必须与 AWS Organizations 中的所有者账户位于同一组织内。
- S3 on Outposts 存储桶拥有者账户始终是存储桶中所有对象的所有者。
- 只有 S3 on Outposts 存储桶拥有者账户可以对存储桶执行操作。
- 对象大小限制与 Amazon S3 一致。
- 在 S3 on Outposts 上存储的所有对象都存储在 OUTPOSTS 存储类中。
- 原定设置情况下，存储在 OUTPOSTS 存储类中的所有对象都使用具有 Amazon S3 托管式加密密钥 (SSE-S3) 的服务器端加密进行存储。您也可以明确选择使用具有客户提供的加密密钥 (SSE-C) 的服务器端加密来存储对象。
- 如果没有足够的空间在您的 Outpost 上存储对象，API 会返回容量不足异常 (ICE)。

S3 on Outposts 支持的 API 操作

有关 S3 on Outposts 支持的 API 操作的列表，请参阅Amazon S3 on Outposts API 操作。

S3 on Outposts 不支持的 Amazon S3 功能

Amazon S3 on Outposts 目前不支持以下 Amazon S3 功能。想要使用它们的任何尝试都将遭到拒绝。

- 访问控制列表 (ACL)
- 跨源资源共享 (CORS)
- S3 分批操作
- S3 清单报告
- 更改默认存储桶加密

- 公有存储桶
- 多重验证 (MFA) 删除
- S3 生命周期转换（除了对象删除和停止未完成的分段上传以外）
- S3 对象锁定依法保留
- 对象锁定保留
- 具有 AWS Key Management Service (AWS KMS) 密钥的服务器端加密 (SSE-KMS)
- S3 复制时间控制 (S3 RTC)
- Amazon CloudWatch 请求指标
- 指标配置
- Transfer Acceleration
- S3 事件通知
- 申请方付款存储桶
- S3 Select
- AWS Lambda 事件
- Server access logging (服务器访问日志记录)
- HTTP POST 请求
- SOAP
- 网站访问

S3 on Outposts 网络要求

- 要将请求路由到 S3 on Outposts 访问点，您必须创建 S3 on Outposts 终端节点并进行配置。以下限制适用于 S3 on Outposts 的终端节点：
  - Outpost 上的每个 Virtual Private Cloud (VPC) 都可以有一个关联的终端节点，每个 Outpost 最多可以有 100 个终端节点。
  - 可以将多个访问点映射到同一终端节点。
  - 只能将终端节点添加到其 CIDR 块在以下 CIDR 范围的子空间内的 VPC 中：
    - 10.0.0.0/8
    - 172.16.0.0/12
    - 192.168.0.0/16
- 只能从具有不重叠 CIDR 块的 VPC 中创建到 Outpost 的终端节点。

- 只能从对应的 Outposts 子网内创建终端节点。
- 用于创建终端节点的子网必须包含四个 IP 地址，以供 S3 on Outposts 使用。
- 如果您指定客户拥有的 IP 地址池（CoIP 池），则它必须包含四个 IP 地址，以供 S3 on Outposts 使用。
- 每个 VPC 每个 Outpost 只能创建一个终端节点。

开始使用 Amazon S3 on Outposts

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 存储桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (OUTPOSTS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud (VPC) 使用访问点和端点连接与 Outposts 存储桶进行通信。您可以像在 Amazon S3 存储桶中一样在 Outpost 存储桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。

通过 Amazon S3 on Outposts，您可以像在 Amazon S3 上一样在 AWS Outposts 上使用 Amazon S3 API 和功能，如对象存储、访问策略、加密和标记。有关 S3 on Outposts 的信息，请参阅什么是 Amazon S3 on Outposts？

主题

- 使用 S3 on Outposts 设置 IAM
- 通过 AWS Management Console开始使用
- 通过 AWS CLI 和适用于 Java 的 SDK 开始使用

使用 S3 on Outposts 设置 IAM

AWS Identity and Access Management（IAM）是一项 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和获得授权（具有权限）来使用 Amazon S3 on Outposts 资源。IAM 是一项无需额外费用即可使用的 AWS 服务。默认情况下，用户对 S3 on Outposts 资源和操作没有权限。要授予对 S3 on Outposts 资源和 API 操作的访问权限，您可以使用 IAM 创建用户、组或角色并附加权限。

要提供访问权限，请为您的用户、组或角色添加权限：

- AWS IAM Identity Center 中的用户和群组：

创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。

- 通过身份提供商在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。
- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

除基于 IAM 身份的策略外，S3 on Outposts 还同时支持桶和接入点策略。存储桶策略和访问点策略是附加到 S3 on Outposts 资源的基于资源的策略。

- 存储桶策略用于附加到存储桶，它根据策略中的元素允许或拒绝对存储桶和存储桶中对象的请求。
- 而访问点策略用于附加到访问点，允许或拒绝对访问点的请求。

访问点策略可与附加到底层 S3 on Outposts 存储桶的存储桶策略结合使用。要使应用程序或用户能够通过某个 S3 on Outposts 访问点访问某个 S3 on Outposts 桶中的对象，访问点策略和桶策略都必须允许该请求。

您在接入点策略中包括的限制仅适用于通过该接入点发出的请求。例如，如果将某个访问点附加到某个桶，则无法使用访问点策略来允许或拒绝直接对该桶发出的请求。但是，应用到存储桶策略的限制可能会允许或拒绝直接对该存储桶或通过该访问点发出的请求。

在 IAM policy 或基于资源的策略中，您可以定义要允许或拒绝的 S3 on Outposts 操作。S3 on Outposts 操作对应于特定的 S3 on Outposts API 操作。S3 on Outposts 操作使用 `s3-outposts:` 命名空间前缀。对 AWS 区域中的 Amazon S3 on Outposts 控制 API 发出的请求，以及对 Outpost 上的对象 API 端点发出的请求，都将使用 IAM 进行身份认证并根据 `s3-outposts:` 命名空间前缀进行授权。要使用 S3 on Outposts，请配置您的 IAM 用户并针对 `s3-outposts:` IAM 命名空间对其进行授权。

有关更多信息，请参阅《服务授权参考》中的 Amazon S3 on Outposts 的操作、资源和条件键。

> **Note**
>
> - S3 on Outposts 不支持访问控制列表 (ACL)。

- S3 on Outposts 默认将存储桶拥有者视为对象所有者，以帮助确保存储桶的拥有者不会被阻止访问或删除对象。
- S3 on Outposts 始终启用“S3 阻止公有访问”，以帮助确保对象从不会具有公有访问权限。

有关为 S3 on Outposts 设置 IAM 的更多信息，请参阅以下主题。

主题

S3 on Outposts 策略的主体

创建基于资源的策略以授予对 S3 on Outposts 存储桶的访问权限时，您必须使用 `Principal` 元素来指定可请求对该资源执行某个操作或运算的人员或应用程序。对于 S3 on Outposts 策略，您可以使用下面的一个主体：

- 一个 AWS 账户
- IAM 用户
- IAM 角色
- 所有主体，通过在使用 `Condition` 元素将访问范围限定为特定 IP 范围的策略中指定通配符 (*) 来确定

Important

您无法为 S3 on Outposts 桶编写在 * 元素中使用通配符（`Principal`）的策略，除非该策略还包含一个将访问范围限制为特定 IP 地址范围的 `Condition`。此限制有助于确保您的 S3 on Outposts 桶不会被公开访问。有关示例，请参阅S3 on Outposts 的示例策略。

有关 `Principal` 元素的更多信息，请参阅《IAM 用户指南》中的 AWS JSON 策略元素：Principal。

S3 on Outposts 的资源 ARN

S3 on Outposts 的 Amazon 资源名称 (ARN) 除了包含 Outpost 归属的 AWS 区域、AWS 账户 ID 和资源名称外，还包含 Outpost ID。要访问 Outposts 存储桶和对象并对它们执行操作，您必须使用下表中显示的一种 ARN 格式。

ARN 中的 *partition* 值是指一组 AWS 区域。每个 AWS 账户 的作用域为一个分区。以下是支持的分区：

- `aws` – AWS 区域
- `aws-us-gov` – AWS GovCloud (US) 区域

S3 on Outposts ARN 格式

Amazon S3 on Outposts ARN	ARN 格式	示例
存储桶 ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /bucket/*bucket_name*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /bucket/*amzn-s3-demo-bucket1*
访问点 ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /accesspoint/ *accesspoint_name*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /accesspoint/ *access-point-name*
对象 ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /bucket/*bucket_name* /object/*object_key*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /bucket/*amzn-s3-demo-*

Amazon S3 on Outposts ARN	ARN 格式	示例
		bucket1 /object/m yobject
S3 on Outposts 访问点对象 ARN（在策略中使用）	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /accesspoint/ *accesspoi nt_name* / object/*object_key*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* : outpost/ *op-01ac5d 28a6a232904* /accesspoint/ *access-point-name/object/myobject*
S3 on Outposts ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* : outpost/ *op-01ac5d 28a6a232904*

S3 on Outposts 的示例策略

Example ：具有 AWS 账户主体的 S3 on Outposts 桶策略

以下存储桶策略使用一个 AWS 账户主体授予对某个 S3 on Outposts 存储桶的访问权限。要使用这一桶策略，请将 *user input placeholders* 替换为您自己的信息。

```
{
   "Version":"2012-10-17",
   "Id":"ExampleBucketPolicy1",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket"
```

```
      }
   ]
}
```

Example ：使用通配符主体 (*) 和条件键将访问范围限定为特定 IP 地址范围的 S3 on Outposts 桶策略

以下桶策略使用通配符主体 (*) 和 aws:SourceIp 条件将访问范围限定为特定的 IP 地址范围。要使用这一桶策略，请将 *user input placeholders* 替换为您自己的信息。

```
{
    "Version": "2012-10-17",
    "Id": "ExampleBucketPolicy2",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": { "AWS" : "*" },
            "Action":"s3-outposts:*",
            "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
                    "aws:SourceIp": "198.51.100.0/24"
                }
            }
        }
    ]
}
```

S3 on Outposts 端点的权限

S3 on Outposts 需要 IAM 中它自己的权限，来管理 S3 on Outposts 端点操作。

> **Note**
>
> - 对于使用客户拥有的 IP 地址池（CoIP 池）访问类型的端点，您还必须具有使用 CoIP 池中 IP 地址的权限，如下表所述。

- 对于使用 AWS Resource Access Manager 访问 S3 on Outposts 的共享账户，这些共享账户中的用户无法在共享子网上创建其自己的端点。如果共享账户中的用户想管理自己的端点，则共享账户必须在 Outpost 上创建自己的子网。有关更多信息，请参阅 the section called “共享 S3 on Outposts”。

S3 on Outposts 端点相关的 IAM 权限

操作	IAM 权限
`CreateEndpoint`	`s3-outposts:CreateEndpoint` `ec2:CreateNetworkInterface` `ec2:DescribeNetworkInterfaces` `ec2:DescribeVpcs` `ec2:DescribeSecurityGroups` `ec2:DescribeSubnets` `ec2:CreateTags` `iam:CreateServiceLinkedRole` 对于使用本地客户拥有的 IP 地址池（CoIP 池）访问类型的端点，将需要以下额外权限： `s3-outposts:CreateEndpoint` `ec2:DescribeCoipPools` `ec2:GetCoipPoolUsage` `ec2:AllocateAddress` `ec2:AssociateAddress` `ec2:DescribeAddresses`

操作	IAM 权限
	ec2:DescribeLocalGatewayRouteTableVpcAssociations
DeleteEndpoint	s3-outposts:DeleteEndpoint ec2:DeleteNetworkInterface ec2:DescribeNetworkInterfaces 对于使用本地客户拥有的 IP 地址池（CoIP 池）访问类型的端点，将需要以下额外权限： s3-outposts:DeleteEndpoint ec2:DisassociateAddress ec2:DescribeAddresses ec2:ReleaseAddress
ListEndpoints	s3-outposts:ListEndpoints

> **Note**
>
> 您可以使用 IAM policy 中的资源标签来管理权限。

S3 on Outposts 的服务相关角色

S3 on Outposts 使用 IAM 服务相关角色代表您创建一些网络资源。有关更多信息，请参阅 将服务相关角色用于 Amazon S3 on Outposts。

通过 AWS Management Console开始使用

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (OUTPOSTS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud（VPC）使用接入点和端点

连接与 Outposts 桶进行通信。您可以像在 Amazon S3 桶中一样在 Outpost 桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅什么是 Amazon S3 on Outposts？

要通过控制台开始使用 S3 on Outposts，请参阅以下主题。要通过 AWS CLI 或AWS SDK for Java 开始使用，请参阅通过 AWS CLI 和适用于 Java 的 SDK 开始使用。

主题

创建存储桶、访问点和端点

以下过程介绍了如何在 S3 on Outposts 中创建第一个存储桶。使用控制台创建存储桶时，您还可以创建一个访问点以及与该存储桶关联的端点，以便您可以立即开始在存储桶中存储对象。

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择创建 Outposts 存储桶。
4. 对于 Bucket name（存储桶名称）中，输入符合域名系统 (DNS) 标准的存储桶名称。

   桶名称必须满足以下要求：

   - 在 AWS 账户、Outpost 以及 Outpost 所属的 AWS 区域内保持唯一性。
   - 长度为 3-63 个字符。
   - 不包含大写字符。
   - 以小写字母或数字开头。

   创建存储桶后，便无法再更改其名称。有关如何命名存储桶的信息，请参阅存储桶命名规则。

   > **Important**
   >
   > 避免在存储桶名称中包含敏感信息，如账号。存储桶名称会显示在指向存储桶中的对象的 URL 中。

5. 对于 Outpost，选择要放置存储桶的 Outpost。
6. 在 Bucket Versioning（桶版本控制）下，将 S3 on Outposts 桶的 S3 版本控制状态设置为以下选项之一：

   - Disable（禁用）（原定设置）- 桶保持未版本控制状态。
   - Enable（启用）- 为桶中的对象启用 S3 版本控制。添加到桶的所有对象都将收到唯一的版本 ID。

   有关 S3 版本控制的更多信息，请参阅为 S3 on Outposts 桶管理 S3 版本控制。
7. （可选）添加要与 Outposts 存储桶关联的任何 optional tags（可选标签）。您可以使用标签来跟踪单个项目或项目组的标准，或使用成本分配标签标注您的存储桶。

   原定设置情况下，存储在 Outposts 存储桶中的所有对象都使用具有 Amazon S3 托管式加密密钥 (SSE-S3) 的服务器端加密进行存储。您也可以明确选择使用具有客户提供的加密密钥 (SSE-C) 的服务器端加密来存储对象。要更改加密类型，您必须使用 REST API、AWS Command Line Interface (AWS CLI) 或 AWS 软件开发工具包。
8. 在 Outposts access point settings（Outposts 访问点设置）部分，输入访问点名称。

   S3 on Outposts 访问点可简化对 S3 on Outposts 中的共享数据集的大规模数据访问管理。访问点是附加到 Outposts 存储桶的已命名网络端点，您可以使用这些存储桶执行 S3 对象操作。有关更多信息，请参阅接入点。

   在此区域和 Outpost 的账户中，访问点名称必须唯一，并符合 接入点限制和局限性。
9. 为此 Amazon S3 on Outposts 访问点选择 VPC。

   如果您没有 VPC，请选择 Create VPC（创建 VPC）。有关更多信息，请参阅创建限制到 Virtual Private Cloud 的接入点。

   通过 Virtual Private Cloud (VPC)，您可以将 AWS 资源启动到您定义的虚拟网络中。这个虚拟网络与您在自己的数据中心中运行的传统网络极其相似，并会为您提供使用的可扩展基础设施的优势 AWS
10. （对于现有 VPC 为可选）为端点选择 Endpoint subnet（端点子网）。

    子网是您的 VPC 内的 IP 地址范围。如果您没有所需的子网，请选择 Create subnet（创建子网）。有关更多信息，请参阅S3 on Outposts 的网络。
11. （对于现有 VPC 为可选）为端点选择 Endpoint security group（端点安全组）。

安全组充当 VPC 的虚拟防火墙，以控制入站和出站流量。

12. （对于现有 VPC 为可选）选择 Endpoint access type（端点访问类型）：
    - Private（私密）– 要与 VPC 一起使用。
    - Customer owned IP（客户拥有的 IP）– 要与本地部署网络中的客户拥有的 IP 池一起使用。
13. （可选）指定 Outpost access point policy（Outpost 访问点策略）。控制台会自动显示访问点的 Amazon Resource Name (ARN)，您可以在策略中使用该名称。
14. 选择创建 Outposts 存储桶。

> **Note**
>
> 创建 Outpost 端点并准备好桶以供使用，最多可能需要 5 分钟的时间。要配置其他存储桶设置，请选择 View details（查看详细信息）。

后续步骤

对于 Amazon S3 on Outposts，对象数据始终存储在 Outpost 上。当 AWS 安装 Outpost 机架时，您的数据将保留在 Outpost 的本地，以满足数据驻留要求。您的对象永远不会离开您的 Outpost，也不在 AWS 区域 中。由于 AWS Management Console 托管在区域内，您无法使用控制台上传或管理 Outpost 中的对象。但是，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 和 AWS SDK 通过访问点上传和管理对象。

创建 S3 on Outposts 存储桶、访问点和端点之后，您可以使用 AWS CLI 或适用于 Java 的 SDK，将对象上传到存储桶。有关更多信息，请参阅将对象上传到 S3 on Outposts 存储桶。

通过 AWS CLI 和适用于 Java 的 SDK 开始使用

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (0UTP0STS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud（VPC）使用接入点和端点连接与 Outposts 桶进行通信。您可以像在 Amazon S3 桶中一样在 Outpost 桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅 什么是 Amazon S3 on Outposts？

要开始使用 S3 on Outposts，您必须创建存储桶、访问点和端点。然后，您可以将对象上传到存储桶。以下示例显示如何通过 AWS CLI 和适用于 Java 的 SDK 开始使用 S3 on Outposts。要通过控制台开始使用，请参阅通过 AWS Management Console开始使用。

主题

步骤 1：创建存储桶

以下 AWS CLI 和适用于 Java 的 SDK 示例显示如何创建 S3 on Outposts 存储桶。

AWS CLI

Example

以下示例使用 AWS CLI 创建 S3 on Outposts 存储桶 (`s3-outposts:CreateBucket`)。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

SDK for Java

Example

以下示例使用适用于 Java 的 SDK 创建 S3 on Outposts 存储桶 (`s3-outposts:CreateBucket`)。

```
import com.amazonaws.services.s3control.model.*;

public String createBucket(String bucketName) {

    CreateBucketRequest reqCreateBucket = new CreateBucketRequest()
            .withBucket(bucketName)
            .withOutpostId(OutpostId)
            .withCreateBucketConfiguration(new CreateBucketConfiguration());
```

```
    CreateBucketResult respCreateBucket =
 s3ControlClient.createBucket(reqCreateBucket);
    System.out.printf("CreateBucket Response: %s%n", respCreateBucket.toString());

    return respCreateBucket.getBucketArn();

}
```

步骤 2：创建访问点

要访问 Amazon S3 on Outposts 存储桶，您必须创建和配置访问点。这些示例显示如何使用 AWS CLI 和适用于 Java 的 SDK 创建访问点。

访问点可简化对 Amazon S3 中的共享数据集的大规模数据访问管理。访问点是附加到存储桶的命名网络端点，您可以使用这些存储桶执行 Amazon S3 对象操作（如 `GetObject` 和 `PutObject`）。对于 S3 on Outposts，您必须使用访问点访问 Outposts 存储桶中的任何对象。访问点仅支持虚拟主机式寻址。

AWS CLI

Example

以下 AWS CLI 示例为 Outposts 存储桶创建访问点。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

SDK for Java

Example

以下适用于 Java 的软件开发工具包示例为 Outpost 存储桶创建访问点。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
import com.amazonaws.services.s3control.model.*;

public String createAccessPoint(String bucketArn, String accessPointName) {
```

```
    CreateAccessPointRequest reqCreateAP = new CreateAccessPointRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withName(accessPointName)
            .withVpcConfiguration(new VpcConfiguration().withVpcId("vpc-12345"));

    CreateAccessPointResult respCreateAP =
 s3ControlClient.createAccessPoint(reqCreateAP);
    System.out.printf("CreateAccessPoint Response: %s%n", respCreateAP.toString());

    return respCreateAP.getAccessPointArn();

}
```

步骤 3：创建端点

要将请求路由到 Amazon S3 on Outposts 访问点，您必须创建 S3 on Outposts 端点并进行配置。为了创建端点，您需要使用服务链接建立到 Outposts 主区域的活跃连接。Outposts 上的每个 Virtual Private Cloud (VPC) 都可以有一个关联的端点。有关端点配额的更多信息，请参阅 S3 on Outposts 网络要求。您必须创建一个端点，才能访问 Outposts 桶并执行对象操作。有关更多信息，请参阅 端点。

这些示例显示如何使用 AWS CLI 和适用于 Java 的 SDK 创建端点。有关创建和管理端点所需的权限的更多信息，请参阅 S3 on Outposts 端点的权限。

AWS CLI

Example

以下 AWS CLI 示例使用 VPC 资源访问类型，为 Outpost 创建端点。VPC 派生自子网。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

以下 AWS CLI 示例使用客户拥有的 IP 地址池（CoIP 池）访问类型为 Outpost 创建端点。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

SDK for Java

Example

以下适用于 Java 的软件开发工具包示例为 Outpost 创建端点。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.CreateEndpointRequest;
import com.amazonaws.services.s3outposts.model.CreateEndpointResult;

public void createEndpoint() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    CreateEndpointRequest createEndpointRequest = new CreateEndpointRequest()
                .withOutpostId("op-0d79779cef3c30a40")
                .withSubnetId("subnet-8c7a57c5")
                .withSecurityGroupId("sg-ab19e0d1")
                .withAccessType("CustomerOwnedIp")
                .withCustomerOwnedIpv4Pool("ipv4pool-coip-12345678901234567");
    // Use .withAccessType and .withCustomerOwnedIpv4Pool only when the access type
 is
    // customer-owned IP address pool (CoIP pool)
    CreateEndpointResult createEndpointResult =
 s3OutpostsClient.createEndpoint(createEndpointRequest);
    System.out.println("Endpoint is created and its ARN is " +
 createEndpointResult.getEndpointArn());
}
```

步骤 4：将对象上传到 S3 on Outposts 存储桶

要上传对象，请参阅将对象上传到 S3 on Outposts 存储桶。

S3 on Outposts 的网络

您可以使用 Amazon S3 on Outposts 为需要本地数据访问、数据处理和数据驻留的应用程序存储和检索本地对象。本节介绍了访问 S3 on Outposts 的网络要求。

主题

选择网络访问类型

您可以从 VPC 内部或本地网络访问 S3 on Outposts。您通过使用访问点和终端节点连接与 Outposts 存储桶进行通信。此连接将在 AWS 网络内的 VPC 和 S3 on Outposts 存储桶之间保持流量。创建终端节点时，您必须将终端节点访问类型指定为 `Private`（对于 VPC 路由）或 `CustomerOwnedIp`（对于客户拥有的 IP 地址池 [CoIP 池]）。

- `Private`（对于 VPC 路由）– 如果您不指定访问类型，S3 on Outposts 将默认使用 `Private`。使用 `Private` 访问类型，VPC 中的实例无需公有 IP 地址便可与 Outpost 中的资源进行通信。您可以从 VPC 内使用 S3 on Outposts。这种类型的端点可以通过直接 VPC 路由从本地网络访问。有关更多信息，请参阅《AWS Outposts 用户指南》中的本地网关路由表。
- `CustomerOwnedIp`（对于 CoIP 池）– 如果您不默认使用 `Private` 访问类型并选择 `CustomerOwnedIp`，则必须指定 IP 地址范围。您可以通过此访问类型从本地网络和 VPC 内使用 S3 on Outposts。访问 VPC 内 Outposts 的 S3 时，您的本地网关的带宽将限制流量。

访问 S3 on Outposts 存储桶和对象

要访问 S3 on Outposts 存储桶和对象，您必须满足以下条件：

- VPC 的访问点
- 同一 VPC 的终端节点
- 您的 Outpost 与 AWS 区域之间的主动连接。有关如何将您的 Outpost 连接到区域的更多信息，请参阅 AWSOutposts 用户指南 中的复制到 Outpost AWS 区域。

有关访问 S3 on Outposts 中的存储桶和对象的更多信息，请参阅使用 S3 on Outposts 存储桶和使用 S3 on Outposts 对象。

跨账户弹性网络接口

S3 on Outposts 终端节点是具有适当 Amazon Resource Name (ARN) 的命名资源。创建这些终端节点后，AWS Outposts 会设置多个跨账户弹性网络接口。S3 on Outposts 跨账户弹性网络接口与其他网络接口相似，但有一个例外：S3 on Outposts 将跨账户弹性网络接口与 Amazon EC2 实例相关联。

S3 on Outposts 域名系统（DNS）通过跨账户弹性网络接口对请求进行负载平衡。S3 on Outposts 在您的AWS账户中创建跨帐户弹性网络接口，它可从 Amazon EC2 控制台的 Network interfaces (网络接口) 窗格中查看。

对于使用 CoIP 池访问类型的终端节点，S3 on Outposts 将从配置的 CoIP 池中分配 IP 地址并将其与跨账户弹性网络接口相关联。

使用 S3 on Outposts 存储桶

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 存储桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (0UTP0STS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您可以像在 Amazon S3 中一样在 Outpost 存储桶上使用相同的 API 和功能，包括访问策略、加密和标记。有关更多信息，请参阅什么是 Amazon S3 on Outposts？

您通过 Virtual Private Cloud (VPC) 使用访问点和终端节点连接与 Outpost 存储桶进行通信。要访问 S3 on Outposts 存储桶和对象，您必须具有 VPC 的访问点和同一 VPC 的终端节点。有关更多信息，请参阅S3 on Outposts 的网络。

桶

在 S3 on Outposts 中，存储桶名称对于 Outposts 是唯一的，并且需要 Outpost 所属区域的 AWS 区域代码，AWS 账户 ID、Outpost ID 和用于识别它们的存储桶名称。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

有关更多信息，请参阅S3 on Outposts 的资源 ARN。

接入点

Amazon S3 on Outposts 支持将纯 Virtual Private Cloud (VPC) 接入点作为访问 Outposts 存储桶的唯一方式。

访问点可简化对 Amazon S3 中的共享数据集的大规模数据访问管理。访问点是附加到存储桶的命名网络端点，您可以使用这些存储桶执行 Amazon S3 对象操作（如 `GetObject` 和 `PutObject`）。对于 S3 on Outposts，您必须使用访问点访问 Outposts 存储桶中的任何对象。访问点仅支持虚拟主机式寻址。

以下示例显示用于 S3 on Outposts 访问点的 ARN 格式。访问点 ARN 包括 Outpost 所属区域的 AWS 区域 代码、AWS 账户 ID、Outpost ID 和访问点名称。

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

端点

要将请求路由到 S3 on Outposts 访问点，您必须创建 S3 on Outposts 终端节点并进行配置。借助 S3 on Outposts 终端节点，您可以将 VPC 私密连接到您的 Outpost 存储桶。S3 on Outposts 终端节点是指向 S3 on Outposts 存储桶的入口点的虚拟统一资源标识符（URI）。它们是水平扩展、冗余和高度可用的 VPC 组件。

Outpost 上的每个 Virtual Private Cloud (VPC) 都可以有一个关联的终端节点，每个 Outpost 最多可以有 100 个终端节点。您必须创建这些终端节点，才能访问 Outpost 存储桶并执行对象操作。创建这些终端节点还通过允许相同的操作在 S3 和 S3 on Outposts 中工作，使 API 模型和行为相同。

S3 on Outposts 上的 API 操作

要管理 Outpost 存储桶 API 操作，S3 on Outposts 将托管一个与 Amazon S3 终端节点不同的单独终端节点。这个终端节点是 `s3-outposts.region.amazonaws.com`。

要使用 Amazon S3 API 操作，您必须使用正确的 ARN 格式，对存储桶和对象进行签署。您必须向 API 操作传递 ARN，以便 Amazon S3 能够确定请求是针对 Amazon S3 (`s3-control.region.amazonaws.com`) 还是针对 S3 on Outposts (`s3-outposts.region.amazonaws.com`)。根据 ARN 格式，然后 S3 可以适当地签署和路由请求。

无论何时将请求发送到 Amazon S3 控制平面，此 SDK 都会从 ARN 中提取组件，并包含一个附加标头 `x-amz-outpost-id`，其 *outpost-id* 值提取自 ARN。ARN 中的服务名称将用于在将请求路由到 S3 on Outposts 终端节点之前对请求进行签名。此行为适用于所有由 `s3control` 客户端处理的所有 API 操作。

下表列出了用于 Amazon S3 on Outposts 的扩展 API 操作及其相对于 Amazon S3 的更改。

API	S3 on Outposts 参数值	
`CreateBucket`	存储桶名称为 ARN，Outpost ID	
`ListRegionalBuckets`	Outpost ID	
`DeleteBucket`	存储桶名称为 ARN	

API	S3 on Outposts 参数值	
DeleteBucketLifecycleConfiguration	存储桶名称为 ARN	
GetBucketLifecycleConfiguration	存储桶名称为 ARN	
PutBucketLifecycleConfiguration	存储桶名称为 ARN	
GetBucketPolicy	存储桶名称为 ARN	
PutBucketPolicy	存储桶名称为 ARN	
DeleteBucketPolicy	存储桶名称为 ARN	
GetBucketTagging	存储桶名称为 ARN	
PutBucketTagging	存储桶名称为 ARN	
DeleteBucketTagging	存储桶名称为 ARN	
CreateAccessPoint	访问点名称为 ARN	
DeleteAccessPoint	访问点名称为 ARN	
GetAccessPoint	访问点名称为 ARN	
GetAccessPoint	访问点名称为 ARN	
ListAccessPoints	访问点名称为 ARN	
PutAccessPointPolicy	访问点名称为 ARN	
GetAccessPointPolicy	访问点名称为 ARN	
DeleteAccessPointPolicy	访问点名称为 ARN	

创建和管理 S3 on Outposts 存储桶

有关创建和管理 S3 on Outposts 存储桶的更多信息，请参阅以下主题。

主题

创建 S3 on Outposts 存储桶

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (OUTPOSTS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud（VPC）使用接入点和端点连接与 Outposts 桶进行通信。您可以像在 Amazon S3 桶中一样在 Outpost 桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅 什么是 Amazon S3 on Outposts？

> **Note**
>
> 创建存储桶的 AWS 账户 拥有该存储桶，也是唯一可以向其提交操作的账户。存储桶具有配置属性，如 Outpost、标签、默认加密和访问点设置。访问点设置包括用于访问存储桶中对象的 Virtual Private Cloud (VPC) 和访问点策略以及其他元数据。有关更多信息，请参阅S3 on Outposts 规范。
>
> 如果您想创建一个桶，该桶使用 AWS PrivateLink 通过虚拟私有云（VPC）中的接口 VPC 端点提供桶和端点管理访问，请参阅适用于 S3 on Outposts 的 AWS PrivateLink。

以下示例显示如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 创建 S3 on Outposts 存储桶。

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择创建 Outposts 存储桶。
4. 对于 Bucket name（存储桶名称）中，输入符合域名系统 (DNS) 标准的存储桶名称。

   桶名称必须满足以下要求：

   - 在 AWS 账户、Outpost 以及 Outpost 所属的 AWS 区域内保持唯一性。
   - 长度为 3-63 个字符。
   - 不包含大写字符。
   - 以小写字母或数字开头。

   创建存储桶后，便无法再更改其名称。有关如何命名存储桶的信息，请参阅存储桶命名规则。

   > **Important**
   >
   > 避免在存储桶名称中包含敏感信息，如账号。存储桶名称会显示在指向存储桶中的对象的 URL 中。

5. 对于 Outpost，选择要放置存储桶的 Outpost。
6. 在 Bucket Versioning（桶版本控制）下，将 S3 on Outposts 桶的 S3 版本控制状态设置为以下选项之一：

   - Disable（禁用）（原定设置）- 桶保持未版本控制状态。
   - Enable（启用）- 为桶中的对象启用 S3 版本控制。添加到桶的所有对象都将收到唯一的版本 ID。

   有关 S3 版本控制的更多信息，请参阅为 S3 on Outposts 桶管理 S3 版本控制。
7. （可选）添加要与 Outposts 存储桶关联的任何 optional tags（可选标签）。您可以使用标签来跟踪单个项目或项目组的标准，或使用成本分配标签标注您的存储桶。

原定设置情况下，存储在 Outposts 存储桶中的所有对象都使用具有 Amazon S3 托管式加密密钥 (SSE-S3) 的服务器端加密进行存储。您也可以明确选择使用具有客户提供的加密密钥 (SSE-C) 的服务器端加密来存储对象。要更改加密类型，您必须使用 REST API、AWS Command Line Interface (AWS CLI) 或 AWS 软件开发工具包。

8. 在 Outposts access point settings（Outposts 访问点设置）部分，输入访问点名称。

   S3 on Outposts 访问点可简化对 S3 on Outposts 中的共享数据集的大规模数据访问管理。访问点是附加到 Outposts 存储桶的已命名网络端点，您可以使用这些存储桶执行 S3 对象操作。有关更多信息，请参阅接入点。

   在此区域和 Outpost 的账户中，访问点名称必须唯一，并符合 接入点限制和局限性。

9. 为此 Amazon S3 on Outposts 访问点选择 VPC。

   如果您没有 VPC，请选择 Create VPC（创建 VPC）。有关更多信息，请参阅创建限制到 Virtual Private Cloud 的接入点。

   通过 Virtual Private Cloud (VPC)，您可以将 AWS 资源启动到您定义的虚拟网络中。这个虚拟网络与您在自己的数据中心中运行的传统网络极其相似，并会为您提供使用的可扩展基础设施的优势 AWS

10. （对于现有 VPC 为可选）为端点选择 Endpoint subnet（端点子网）。

    子网是您的 VPC 内的 IP 地址范围。如果您没有所需的子网，请选择 Create subnet（创建子网）。有关更多信息，请参阅S3 on Outposts 的网络。

11. （对于现有 VPC 为可选）为端点选择 Endpoint security group（端点安全组）。

    安全组充当 VPC 的虚拟防火墙，以控制入站和出站流量。

12. （对于现有 VPC 为可选）选择 Endpoint access type（端点访问类型）：

    - Private（私密）– 要与 VPC 一起使用。
    - Customer owned IP（客户拥有的 IP）– 要与本地部署网络中的客户拥有的 IP 池一起使用。

13. （可选）指定 Outpost access point policy（Outpost 访问点策略）。控制台会自动显示访问点的 Amazon Resource Name (ARN)，您可以在策略中使用该名称。

14. 选择创建 Outposts 存储桶。

> **Note**
>
> 创建 Outpost 端点并准备好桶以供使用，最多可能需要 5 分钟的时间。要配置其他存储桶设置，请选择 View details（查看详细信息）。

使用 AWS CLI

Example

以下示例使用 AWS CLI 创建 S3 on Outposts 存储桶 (`s3-outposts:CreateBucket`)。要运行此命令，请将 *`user input placeholders`* 替换为您自己的信息。

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

使用适用于 Java 的 AWS 软件开发工具包

Example

以下示例使用适用于 Java 的 SDK 创建 S3 on Outposts 存储桶 (`s3-outposts:CreateBucket`)。

```
import com.amazonaws.services.s3control.model.*;

public String createBucket(String bucketName) {

    CreateBucketRequest reqCreateBucket = new CreateBucketRequest()
            .withBucket(bucketName)
            .withOutpostId(OutpostId)
            .withCreateBucketConfiguration(new CreateBucketConfiguration());

    CreateBucketResult respCreateBucket =
 s3ControlClient.createBucket(reqCreateBucket);
    System.out.printf("CreateBucket Response: %s%n", respCreateBucket.toString());

    return respCreateBucket.getBucketArn();

}
```

为 S3 on Outposts 存储桶添加标签

您可以为 Amazon S3 on Outposts 存储桶添加标签，以跟踪各个项目或项目组的存储成本和其他标准。

> **Note**
>
> 创建存储桶的 AWS 账户拥有该存储桶，也是唯一可以更改其标签的账户。

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择您要编辑其标签的 Outposts 存储桶。
4. 选择 Properties（属性）选项卡。
5. 在 Tags（标签）下，选择 Edit（编辑）。
6. 选择 Add new tag（添加新标签），然后输入 Key（键）和可选的 Value（值）。

   添加您想要与 Outposts 存储桶关联的任何标签，以跟踪单个项目或项目组的其他标准。
7. 选择 Save changes（保存更改）。

使用 AWS CLI

以下 AWS CLI 示例使用当前文件夹中指定标签的 JSON 文档 (*tagging.json*) 将标记配置应用于 S3 on Outposts 存储桶。要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging file://tagging.json

tagging.json

{
   "TagSet": [
     {
       "Key": "organization",
```

```
        "Value": "marketing"
      }
    ]
}
```

以下 AWS CLI 示例直接从命令行将标签配置应用于 S3 on Outposts 存储桶。

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging 'TagSet=[{Key=organization,Value=marketing}]'
```

有关此命令的更多信息，请参阅《AWS CLI 参考》中的 put-bucket-tagging。

使用存储桶策略管理对 Amazon S3 on Outposts 存储桶的访问

存储桶策略是基于资源的 AWS Identity and Access Management (IAM) 策略，您可以使用该策略向存储桶及其中对象授予访问权限。只有存储桶拥有者才能将策略与存储桶关联。附加到存储桶的权限适用于存储桶拥有者拥有的存储桶中所有对象。存储桶策略的大小限制为 20 KB。有关更多信息，请参阅存储桶策略。

您可以更新存储桶策略来管理对 Amazon S3 on Outposts 存储桶的访问。有关更多信息，请参阅以下主题。

主题

- 为 Amazon S3 on Outposts 存储桶添加或编辑存储桶策略
- 查看 Amazon S3 on Outposts 存储桶策略
- 删除 Amazon S3 on Outposts 存储桶策略
- 存储桶策略示例

为 Amazon S3 on Outposts 存储桶添加或编辑存储桶策略

存储桶策略是基于资源的 AWS Identity and Access Management (IAM) 策略，您可以使用该策略向存储桶及其中对象授予访问权限。只有存储桶所有者才能将策略与存储桶关联。附加到存储桶的权限适用于存储桶所有者拥有的存储桶中所有对象。存储桶策略的大小限制为 20 KB。有关更多信息，请参阅存储桶策略。

以下主题显示如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS SDK for Java 更新 Amazon S3 on Outposts 存储桶策略。

使用 S3 控制台

创建或编辑存储桶策略

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择您要编辑其权限的 Outposts 存储桶。
4. 请选择 Permissions 选项卡。
5. 在 Outposts bucket policy（Outposts 存储桶策略）部分，要创建或编辑新策略，请选择 Edit（编辑）。

   现在，您可以添加或编辑 S3 on Outposts 存储桶策略。有关更多信息，请参阅使用 S3 on Outposts 设置 IAM。

使用 AWS CLI

以下 AWS CLI 示例在 Outpost 存储桶上放置策略。

1. 将以下存储桶策略保存到 JSON 文件中。在本示例中，文件命名为 `policy1.json`。将 *`user input placeholders`* 替换为您自己的信息。

```
{
   "Version":"2012-10-17",
   "Id":"testBucketPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket"
      }
   ]
}
```

2. 将 JSON 文件作为 put-bucket-policy CLI 命令的一部分提交。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control put-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket --policy file://policy1.json
```

使用适用于 Java 的 AWS 软件开发工具包

以下 SDK for Java 示例在 Outpost 存储桶上放置策略。

```
import com.amazonaws.services.s3control.model.*;

public void putBucketPolicy(String bucketArn) {

    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testBucketPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId+ "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + bucketArn + "\"}]}";

    PutBucketPolicyRequest reqPutBucketPolicy = new PutBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withPolicy(policy);

    PutBucketPolicyResult respPutBucketPolicy =
 s3ControlClient.putBucketPolicy(reqPutBucketPolicy);
    System.out.printf("PutBucketPolicy Response: %s%n",
 respPutBucketPolicy.toString());

}
```

查看 Amazon S3 on Outposts 存储桶策略

存储桶策略是基于资源的 AWS Identity and Access Management (IAM) 策略，您可以使用该策略向存储桶及其中对象授予访问权限。只有存储桶所有者才能将策略与存储桶关联。附加到存储桶的权限适用于存储桶所有者拥有的存储桶中所有对象。存储桶策略的大小限制为 20 KB。有关更多信息，请参阅存储桶策略。

以下主题说明如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS SDK for Java 查看 Amazon S3 on Outposts 存储桶策略。

使用 S3 控制台

创建或编辑存储桶策略

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择您要编辑其权限的 Outposts 存储桶。
4. 选择 Permissions 选项卡。
5. 在 Outposts bucket policy（Outposts 存储桶策略）部分，您可以查看现有的存储桶策略。有关更多信息，请参阅使用 S3 on Outposts 设置 IAM。

使用 AWS CLI

以下 AWS CLI 示例获取 Outposts 存储桶的策略。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket
```

使用适用于 Java 的 AWS 软件开发工具包

以下适用于 Java 的开发工具包示例为 Outpost 存储桶获取策略。

```
import com.amazonaws.services.s3control.model.*;

public void getBucketPolicy(String bucketArn) {

    GetBucketPolicyRequest reqGetBucketPolicy = new GetBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    GetBucketPolicyResult respGetBucketPolicy =
 s3ControlClient.getBucketPolicy(reqGetBucketPolicy);
    System.out.printf("GetBucketPolicy Response: %s%n",
 respGetBucketPolicy.toString());

}
```

删除 Amazon S3 on Outposts 存储桶策略

存储桶策略是基于资源的 AWS Identity and Access Management (IAM) 策略，您可以使用该策略向存储桶及其中对象授予访问权限。只有存储桶所有者才能将策略与存储桶关联。附加到存储桶的权限适用于存储桶所有者拥有的存储桶中所有对象。存储桶策略的大小限制为 20 KB。有关更多信息，请参阅存储桶策略。

以下主题显示如何使用 AWS Management Console或 AWS Command Line Interface (AWS CLI) 查看 Amazon S3 on Outposts 存储桶策略。

使用 S3 控制台

删除存储桶策略

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择您要编辑其权限的 Outposts 存储桶。
4. 选择 Permissions 选项卡。
5. 在 Outposts bucket policy（Outposts 存储桶策略）部分中，选择 Delete（删除）。
6. 确认删除操作。

使用 AWS CLI

以下示例使用 AWS CLI 删除 S3 on Outposts 存储桶策略 (`s3-outposts:DeleteBucket`)。要运行此命令，请将 *`user input placeholders`* 替换为您自己的信息。

```
aws s3control delete-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

存储桶策略示例

使用 S3 on Outposts 桶策略，您可以保护对 S3 on Outposts 桶中对象的访问，这样，只有具有适当权限的用户才能访问它们。您甚至可以阻止没有适当权限的经过身份验证的用户访问您的 S3 on Outposts 资源。

本节介绍关于 S3 on Outposts 桶策略的典型使用案例的示例。要测试这些策略，您需要将 *`user input placeholders`* 替换为您自己的信息（例如存储桶名称）。

要授予或拒绝对一组对象的权限，可以在 Amazon资源名称（ARN）和其他值中使用通配符（*）。例如，您可以控制对以通用前缀或以给定扩展名结尾的对象组的访问，例如 .html。

有关 AWS Identity and Access Management（IAM）策略语言的更多信息，请参阅使用 S3 on Outposts 设置 IAM。

> **Note**
>
> 使用 Amazon S3 控制台测试 s3outposts 权限时，您必须授予控制台所需的其它权限，如 s3outposts:createendpoint、s3outposts:listendpoints 等等。

用于创建存储桶策略的其他资源

- 有关在创建 S3 on Outposts 桶策略时可以使用的 IAM policy 操作、资源和条件键的列表，请参阅 Actions, resources, and condition keys for Amazon S3 on Outposts。
- 有关创建 S3 on Outposts 策略的指南，请参阅为 Amazon S3 on Outposts 存储桶添加或编辑存储桶策略。

主题

- 基于特定 IP 地址管理对 Amazon S3 on Outposts 桶的访问权限

基于特定 IP 地址管理对 Amazon S3 on Outposts 桶的访问权限

存储桶策略是基于资源的 AWS Identity and Access Management (IAM) 策略，您可以使用该策略向存储桶及其中对象授予访问权限。只有存储桶拥有者才能将策略与存储桶关联。附加到存储桶的权限适用于存储桶拥有者拥有的存储桶中所有对象。存储桶策略的大小限制为 20 KB。有关更多信息，请参阅存储桶策略。

限制特定 IP 地址的访问权限

以下示例拒绝所有用户对指定桶中的对象执行任何 S3 on Outposts 操作，除非请求来自指定的 IP 地址范围。

> **Note**
>
> 在限制对特定 IP 地址的访问权限时，务必还要指定哪些 VPC 端点、VPC 源 IP 地址或外部 IP 地址可以访问 S3 on Outposts 桶。否则，如果您的策略拒绝所有用户在没有适当权限的情况

下对 S3 on Outposts 桶中的对象执行任何 s3outposts 操作，则您可能会失去对该桶的访问权限。

此策略的 Condition 语句确定允许的 IP 版本 4（IPv4）IP 地址范围为 *192.0.2.0/24*。

Condition 块使用 NotIpAddress 条件和 aws:SourceIp 条件键（这是 AWS 范围的条件键）。aws:SourceIp 条件键只能用于公有 IP 地址范围。有关条件键的更多信息，请参阅 Actions, resources, and condition keys for S3 on Outposts。aws:SourceIp IPv4 值使用标准 CIDR 表示法。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

Warning

在使用此 S3 on Outposts 策略之前，将此示例中的 *192.0.2.0/24* IP 地址范围替换为适合您的使用案例的值。否则，您将失去访问桶的能力。

```
{
    "Version": "2012-10-17",
    "Id": "S3OutpostsPolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3outposts:*",
            "Resource": [
                "arn:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
accesspoint/EXAMPLE-ACCESS-POINT-NAME"
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET"
            ],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

同时允许 IPv4 和 IPv6 地址

在您开始使用 IPv6 地址时，建议您使用您的 IPv6 地址范围以及现有的 IPv4 范围来更新组织的所有策略。这样做将有助于确保这些策略在您转换到 IPv6 时继续有效。

以下 S3 on Outposts 示例桶策略说明如何混用 IPv4 和 IPv6 地址范围来覆盖组织的所有有效 IP 地址。该示例策略将允许对示例 IP 地址 *192.0.2.1* 和 *2001:DB8:1234:5678::1* 的访问，而拒绝对地址 *203.0.113.1* 和 *2001:DB8:1234:5678:ABCD::1* 的访问。

aws:SourceIp 条件键只能用于公有 IP 地址范围。aws:SourceIp 的 IPv6 值必须采用标准的 CIDR 格式。对于 IPv6，我们支持使用 :: 表示 0 范围（例如，2001:DB8:1234:5678::/64）。有关更多信息，请参阅《IAM 用户指南》中的 IP 地址条件运算符。

> **Warning**
>
> 在使用此 S3 on Outposts 策略之前，将此示例中的 IP 地址范围替换为适合您的使用案例的值。否则，您可能会失去访问存储桶的能力。

```
{
    "Id": "S3OutpostsPolicyId2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIPmix",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3outposts:*",
            "Resource": [
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET",
                        "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-
ID/bucket/DOC-EXAMPLE-BUCKET/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                },
                "NotIpAddress": {
```

```
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678:ABCD::/80"
                    ]
                }
            }
        }
    ]
}
```

列出 Amazon S3 on Outposts 存储桶

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 存储桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (0UTP0STS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud (VPC) 使用访问点和端点连接与 Outposts 存储桶进行通信。您可以像在 Amazon S3 存储桶中一样在 Outpost 存储桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅什么是 Amazon S3 on Outposts？

有关使用 S3 on Outposts 中的存储桶的更多信息，请参阅使用 S3 on Outposts 存储桶。

以下示例介绍了如何使用 AWS Management Console、AWS CLI 和 AWS SDK for Java 返回 S3 on Outposts 存储桶的列表。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 在 Outposts buckets（Outposts 存储桶）下，查看 S3 on Outposts 存储桶的列表。

使用 AWS CLI

以下 AWS CLI 示例获取 Outpost 中的存储桶列表。要使用此命令，请将每个 *user input placeholder* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 list-regional-buckets。

```
aws s3control list-regional-buckets --account-id 123456789012 --outpost-
id op-01ac5d28a6a232904
```

使用适用于 Java 的 AWS SDK

以下适用于 Java 的 SDK 示例获取 Outpost 中的存储桶列表。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 ListRegionalBuckets。

```
import com.amazonaws.services.s3control.model.*;

public void listRegionalBuckets() {

    ListRegionalBucketsRequest reqListBuckets = new ListRegionalBucketsRequest()
            .withAccountId(AccountId)
            .withOutpostId(OutpostId);

    ListRegionalBucketsResult respListBuckets =
 s3ControlClient.listRegionalBuckets(reqListBuckets);
    System.out.printf("ListRegionalBuckets Response: %s%n",
 respListBuckets.toString());

}
```

使用 AWS CLI 和适用于 Java 的 SDK 获取 S3 on Outposts 存储桶

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 存储桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (0UTP0STS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud (VPC) 使用访问点和端点连接与 Outposts 存储桶进行通信。您可以像在 Amazon S3 存储桶中一样在 Outpost 存储桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅什么是 Amazon S3 on Outposts？

以下示例显示了如何使用 AWS CLI 和 AWS SDK for Java 获取 S3 on Outposts 存储桶。

> **Note**
>
> 当您通过 AWS CLI 或 AWS SDK 使用 Amazon S3 on Outposts 时，您可以提供 Outpost 的访问点 ARN 来代替存储桶名称。访问点 ARN 采用以下形式，其中 *region* 是 Outpost 归属的区域的 AWS 区域代码：
>
> `arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-access-point`

有关 S3 on Outposts ARN 的更多信息，请参阅S3 on Outposts 的资源 ARN。

使用 AWS CLI

以下 S3 on Outposts 示例使用 AWS CLI 获取存储桶。要使用此命令，请将每个 *user input placeholder* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 get-bucket。

```
aws s3control get-bucket --account-id 123456789012 --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket"
```

使用适用于 Java 的 AWS SDK

以下 S3 on Outposts 示例使用适用于 Java 的 SDK 获取存储桶。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 GetBucket。

```
import com.amazonaws.services.s3control.model.*;

public void getBucket(String bucketArn) {

    GetBucketRequest reqGetBucket = new GetBucketRequest()
            .withBucket(bucketArn)
            .withAccountId(AccountId);

    GetBucketResult respGetBucket = s3ControlClient.getBucket(reqGetBucket);
    System.out.printf("GetBucket Response: %s%n", respGetBucket.toString());

}
```

删除 Amazon S3 on Outposts 存储桶

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 存储桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (0UTP0STS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud (VPC) 使用访问点和端点连接与 Outposts 存储桶进行通信。您可以像在 Amazon S3 存储桶中一样在 Outpost 存储桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS

Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅什么是 Amazon S3 on Outposts？

有关使用 S3 on Outposts 中的存储桶的更多信息，请参阅使用 S3 on Outposts 存储桶。

创建存储桶的 AWS 账户 拥有该存储桶，也是唯一可以删除它的账户。

> **Note**
>
> - Outposts 存储桶必须为空才能进行删除。
>
> Amazon S3 控制台不支持 S3 on Outposts 对象操作。要删除 S3 on Outposts 存储桶中的对象，您可以使用 AWS CLI、AWS 软件开发工具包或 REST API。
> - 在删除 Outposts 存储桶之前，必须先删除该存储桶的任何 Outposts 访问点。有关更多信息，请参阅删除访问点。
> - 存储桶在删除后不能恢复。

以下示例显示如何使用 AWS Management Console和 AWS Command Line Interface (AWS CLI) 删除 S3 on Outposts 存储桶。

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择要删除的存储桶，然后选择 Delete（删除）。
4. 确认删除操作。

使用 AWS CLI

以下示例使用 AWS CLI 删除 S3 on Outposts 存储桶 (`s3-outposts:DeleteBucket`)。要运行此命令，请将 *`user input placeholders`* 替换为您自己的信息。

```
aws s3control delete-bucket --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

使用 Amazon S3 on Outposts 访问点

要访问 Amazon S3 on Outposts 存储桶，您必须创建和配置访问点。

访问点可简化对 Amazon S3 中的共享数据集的大规模数据访问管理。访问点是附加到存储桶的命名网络端点，您可以使用这些存储桶执行 Amazon S3 对象操作（如 `GetObject` 和 `PutObject`）。对于 S3 on Outposts，您必须使用访问点访问 Outposts 存储桶中的任何对象。访问点仅支持虚拟主机式寻址。

> **Note**
>
> 创建 Outposts 存储桶的 AWS 账户 拥有该存储桶，也是唯一可以为其分配访问点的账户。

以下部分介绍如何创建和管理 S3 on Outposts 存储桶的访问点。

主题

创建 S3 on Outposts 访问点

要访问 Amazon S3 on Outposts 存储桶，您必须创建和配置访问点。

访问点可简化对 Amazon S3 中的共享数据集的大规模数据访问管理。访问点是附加到存储桶的命名网络端点，您可以使用这些存储桶执行 Amazon S3 对象操作（如 `GetObject` 和 `PutObject`）。对于 S3 on Outposts，您必须使用访问点访问 Outposts 存储桶中的任何对象。访问点仅支持虚拟主机式寻址。

以下示例显示如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 创建 S3 on Outposts 访问点。

> **Note**
>
> 创建 Outposts 存储桶的 AWS 账户 拥有该存储桶，也是唯一可以为其分配访问点的账户。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择要为其创建 Outposts 访问点的 Outposts 存储桶。
4. 选择 Outposts 访问点选项卡。
5. 在 Outposts access points（Outposts 访问点）部分中，选择 Create Outposts access point（创建 Outposts 访问点）。
6. 在 Outposts access point settings（Outposts 访问点设置）中，输入访问点的名称，然后选择访问点的 Virtual Private Cloud (VPC)。
7. 如果要为访问点添加策略，请在 Outposts access point policy（Outposts 访问点策略）部分中输入策略。

   有关更多信息，请参阅使用 S3 on Outposts 设置 IAM。

使用 AWS CLI

Example

以下 AWS CLI 示例为 Outposts 存储桶创建访问点。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

使用适用于 Java 的 AWS 软件开发工具包

Example

以下适用于 Java 的软件开发工具包示例为 Outpost 存储桶创建访问点。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
import com.amazonaws.services.s3control.model.*;

public String createAccessPoint(String bucketArn, String accessPointName) {

    CreateAccessPointRequest reqCreateAP = new CreateAccessPointRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withName(accessPointName)
            .withVpcConfiguration(new VpcConfiguration().withVpcId("vpc-12345"));

    CreateAccessPointResult respCreateAP =
 s3ControlClient.createAccessPoint(reqCreateAP);
    System.out.printf("CreateAccessPoint Response: %s%n", respCreateAP.toString());

    return respCreateAP.getAccessPointArn();

}
```

为您的 S3 on Outposts 桶访问点使用桶式别名

对于 S3 on Outposts，您必须使用接入点访问 Outposts 桶中的任何对象。每次您为桶创建接入点时，S3 on Outposts 都会自动生成一个接入点别名。对于任何数据面板操作，您都可以使用此访问点别名，而不是访问点 ARN。例如，您可以使用访问点别名来执行对象级操作，例如 PUT、GET、LIST 等。有关这些操作的列表，请参阅用于管理对象的 Amazon S3 API 操作。

以下示例显示了名为 *my-access-point* 的访问点的 ARN 和访问点别名。

- 访问点 ARN – `arn:aws:s3-outposts:`*region*`:`*123456789012*`:outpost/`*op-01ac5d28a6a232904*`/accesspoint/`*my-access-point*
- 访问点别名 – *my-access-po-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10*`--op-s3`

有关 ARN 的更多信息，请参阅《AWS 一般参考》中的 Amazon 资源名称（ARN）。

有关访问点别名的更多信息，请参阅以下主题：

主题

- 访问点别名
- 在 S3 on Outposts 对象操作中使用访问点别名
- 限制

访问点别名

访问点别名是在与 S3 on Outposts 桶相同的命名空间中创建的。当您创建访问点时，S3 on Outposts 会自动生成一个无法更改的访问点别名。访问点别名符合有效 S3 on Outposts 桶名称的所有要求，包括以下部分：

access point name prefix-*metadata*--op-s3

> **Note**
>
> --op-s3 后缀保留用于访问点别名，我们建议您不要将其用于桶或访问点名称。有关 S3 on Outposts 桶命名规则的更多信息，请参阅使用 S3 on Outposts 存储桶。

查找访问点别名

以下示例显示了如何使用 AmazonS3 控制台和 AWS CLI 查找访问点别名。

Example ：在 Amazon S3 控制台中查找并复制访问点别名

在控制台中创建访问点后，可以从 Access Points（访问点）列表的 Access Point alias（访问点别名）列中获取访问点别名。

复制访问点别名

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts access points（Outposts 访问点）。
3. 要复制访问点别名，请执行以下操作之一：
   - 在 Access Points（访问点）列表中，选择访问点名称旁边的选项按钮，然后选择 Copy Access Point alias（复制访问点别名）。
   - 请选择接入点名称。然后，在 Outposts access point overview（Outposts 访问点概述）下，复制访问点别名。

Example ：使用 AWS CLI 创建访问点并在响应中查找访问点别名

以下有关 create-access-point 命令的 AWS CLI 示例创建访问点并返回自动生成的访问点别名。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-access-point --bucket example-outposts-bucket --name example-
outposts-access-point --account-id 123456789012

{
    "AccessPointArn":
    "arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
accesspoint/example-outposts-access-point",
    "Alias": "example-outp-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

Example ：使用 AWS CLI 获取访问点别名

例如，以下有关 `get-access-point` 命令的 AWS CLI 示例返回关于指定的访问点的信息。此信息包括访问点别名。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-access-point --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --name example-outposts-access-point --account-id 123456789012

{
    "Name": "example-outposts-access-point",
    "Bucket": "example-outposts-bucket",
    "NetworkOrigin": "Vpc",
    "VpcConfiguration": {
        "VpcId": "vpc-01234567890abcdef"
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2022-09-18T17:49:15.584000+00:00",
    "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

Example ：使用 AWS CLI 列出访问点以查找访问点别名

例如，以下有关 `list-access-points` 命令的 AWS CLI 示例列出关于指定的访问点的信息。此信息包括访问点别名。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket

{
    "AccessPointList": [
        {
            "Name": "example-outposts-access-point",
            "NetworkOrigin": "Vpc",
            "VpcConfiguration": {
                "VpcId": "vpc-01234567890abcdef"
            },
            "Bucket": "example-outposts-bucket",
            "AccessPointArn": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point",
            "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
        }
    ]
}
```

在 S3 on Outposts 对象操作中使用访问点别名

采用访问点时，您可以使用访问点别名，而无需对代码进行大量更改。

此 AWS CLI 示例显示了针对 S3 on Outposts 桶的 `get-object` 操作。此示例使用访问点别名作为 `--bucket` 的值，而不是完整的访问点 ARN。

```
aws s3api get-object --bucket my-access-po-
o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3 --key  testkey  sample-object.rtf

{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
}
```

限制

- 客户无法配置别名。
- 无法在访问点上删除、修改或禁用别名。
- 您不能将访问点别名用于 S3 on Outposts 控制面板操作。有关 S3 on Outposts 控制面板操作的列表，请参阅用于管理存储桶的 Amazon S3 控制 API。
- 别名不能用于 AWS Identity and Access Management（IAM）策略中。

查看有关访问点配置的信息

访问点可简化对 Amazon S3 中的共享数据集的大规模数据访问管理。访问点是附加到存储桶的命名网络终端节点，您可以使用这些存储桶执行 Amazon S3 对象操作（如 `GetObject` 和 `PutObject`）。对于 S3 on Outposts，您必须使用访问点访问 Outposts 存储桶中的任何对象。接入点仅支持虚拟主机式寻址。

以下主题说明如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 返回 S3 on Outposts 访问点的配置信息。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts access points（Outposts 访问点）。
3. 选择要查看其配置详细信息的 Outposts 访问点。
4. 在 Outposts access point overview（Outposts 访问点概述）下，查看访问点配置详细信息。

使用 AWS CLI

以下 AWS CLI 示例获取 Outpost 存储桶的访问点。将 *`user input placeholders`* 替换为您自己的信息。

```
aws s3control get-access-point --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

使用适用于 Java 的 AWS 软件开发工具包

以下适用于 Java 的软件开发工具包示例为 Outpost 存储桶获取访问点。

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPoint(String accessPointArn) {

    GetAccessPointRequest reqGetAP = new GetAccessPointRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointResult respGetAP = s3ControlClient.getAccessPoint(reqGetAP);
    System.out.printf("GetAccessPoint Response: %s%n", respGetAP.toString());

}
```

查看 Amazon S3 on Outposts 访问点的列表

访问点可简化对 Amazon S3 中的共享数据集的大规模数据访问管理。访问点是附加到存储桶的命名网络端点，您可以使用这些存储桶执行 Amazon S3 对象操作（如 `GetObject` 和 `PutObject`）。对于 S3 on Outposts，您必须使用访问点访问 Outposts 存储桶中的任何对象。访问点仅支持虚拟主机式寻址。

以下主题说明如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 返回 S3 on Outposts 访问点的列表。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts access points（Outposts 访问点）。
3. 在 Outposts access points（Outposts 访问点）下，查看您的 S3 on Outposts 访问点列表。

使用 AWS CLI

以下 AWS CLI 示例列出 Outposts 存储桶的接入点。要运行此命令，请将 *`user input placeholders`* 替换为您自己的信息。

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

使用适用于 Java 的 AWS 软件开发工具包

以下 SDK for Java 示例列出 Outposts 存储桶的接入点。

```
import com.amazonaws.services.s3control.model.*;

public void listAccessPoints(String bucketArn) {

    ListAccessPointsRequest reqListAPs = new ListAccessPointsRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    ListAccessPointsResult respListAPs = s3ControlClient.listAccessPoints(reqListAPs);
    System.out.printf("ListAccessPoints Response: %s%n", respListAPs.toString());

}
```

删除访问点

访问点可简化对 Amazon S3 中的共享数据集的大规模数据访问管理。访问点是附加到存储桶的命名网络终端节点，您可以使用这些存储桶执行 Amazon S3 对象操作（如 `GetObject` 和 `PutObject`）。对于 S3 on Outposts，您必须使用访问点访问 Outposts 存储桶中的任何对象。接入点仅支持虚拟主机式寻址。

以下示例显示如何使用 AWS Management Console和 AWS Command Line Interface (AWS CLI) 删除访问点。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts access points（Outposts 访问点）。
3. 在 Outposts access points（Outposts 访问点）部分中，选择要删除的 Outposts 访问点。
4. 选择 Delete。
5. 确认删除操作。

使用 AWS CLI

以下 AWS CLI 示例删除 Outposts 访问点。要运行此命令，请将 *`user input placeholders`* 替换为您自己的信息。

```
aws s3control delete-access-point --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

添加或编辑访问点策略

访问点具有不同的权限和网络控制，Amazon S3 on Outposts 将它们应用于通过该访问点发出的任何请求。每个访问点强制实施自定义访问点策略，该策略与附加到底层存储桶的存储桶策略结合使用。有关更多信息，请参阅接入点。

以下主题说明了如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 添加或编辑您的 S3 on Outposts 访问点策略。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择要为其编辑访问点策略的 Outposts 存储桶。
4. 选择 Outposts 访问点选项卡。
5. 在 Outposts access points（Outposts 访问点）部分中，选择要编辑其策略的访问点，然后选择 Edit policy（编辑策略）。
6. 在 Outposts access point policy（Outposts 访问点策略）部分中添加或编辑策略。有关更多信息，请参阅使用 S3 on Outposts 设置 IAM。

使用 AWS CLI

以下 AWS CLI 示例在 Outposts 访问点上放置策略。

1. 将以下访问点策略保存到 JSON 文件中。在本示例中，文件命名为 `appolicy1.json`。将 *`user input placeholders`* 替换为您自己的信息。

```
{
   "Version":"2012-10-17",
   "Id":"exampleAccessPointPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
```

```
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point
      }
   ]
}
```

2. 将 JSON 文件作为 put-access-point-policy CLI 命令的一部分提交。将 *user input placeholders* 替换为您自己的信息。

```
aws s3control put-access-point-policy --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --policy file://appolicy1.json
```

使用适用于 Java 的 AWS 软件开发工具包

以下 SDK for Java 示例在 Outposts 接入点上放置策略。

```
import com.amazonaws.services.s3control.model.*;

public void putAccessPointPolicy(String accessPointArn) {

    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testAccessPointPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId + "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + accessPointArn +
 "\"}]}";

    PutAccessPointPolicyRequest reqPutAccessPointPolicy = new
 PutAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn)
            .withPolicy(policy);

    PutAccessPointPolicyResult respPutAccessPointPolicy =
 s3ControlClient.putAccessPointPolicy(reqPutAccessPointPolicy);
    System.out.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());
    printWriter.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());
```

```
}
```

查看 S3 on Outposts 访问点策略

访问点具有不同的权限和网络控制，Amazon S3 on Outposts 将它们应用于通过该访问点发出的任何请求。每个访问点强制实施自定义访问点策略，该策略与附加到底层存储桶的存储桶策略结合使用。有关更多信息，请参阅接入点。

有关使用 S3 on Outposts 中的访问点的更多信息，请参阅使用 S3 on Outposts 存储桶。

以下主题说明如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 查看您的 S3 on Outposts 访问点策略。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts access points（Outposts 访问点）。
3. 选择要查看其策略的 Outposts 访问点。
4. 在 Permissions（权限）选项卡上，查看 S3 on Outposts 访问点策略。
5. 要编辑访问点策略，请参阅添加或编辑访问点策略。

使用 AWS CLI

以下 AWS CLI 示例获取 Outposts 接入点策略。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-access-point-policy --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

使用适用于 Java 的 AWS 软件开发工具包

以下 SDK for Java 示例获取 Outposts 接入点策略。

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPointPolicy(String accessPointArn) {
```

```
    GetAccessPointPolicyRequest reqGetAccessPointPolicy = new
 GetAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointPolicyResult respGetAccessPointPolicy =
 s3ControlClient.getAccessPointPolicy(reqGetAccessPointPolicy);
    System.out.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());
    printWriter.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());

}
```

使用 Amazon S3 on Outposts 端点

要将请求路由到 Amazon S3 on Outposts 访问点，您必须创建并配置 S3 on Outposts 端点。为了创建端点，您需要使用服务链接建立到 Outposts 主区域的活跃连接。Outposts 上的每个 Virtual Private Cloud (VPC) 都可以有一个关联的端点。有关端点配额的更多信息，请参阅 S3 on Outposts 网络要求。您必须创建一个端点，才能访问 Outposts 桶并执行对象操作。有关更多信息，请参阅端点。

创建端点后，您可以使用“状态”字段来了解端点的状态。如果您的 Outposts 处于离线状态，它将返回 CREATE_FAILED。您可以检查服务链接连接，删除端点，并在连接恢复后重试创建操作。有关其他错误代码的列表，请参阅下文。有关更多信息，请参阅端点。

API	状态	失败原因错误代码	消息 - 失败原因
`CreateEndpoint`	Create_Failed	OutpostNotReachable	由于与您的 Outposts 主区域的服务链接连接已断开，因此无法创建端点。请检查您的连接，删除端点，然后重试。
`CreateEndpoint`	Create_Failed	InternalError	由于内部错误，无法创建端点。请删除端点并重新创建。
`DeleteEndpoint`	Delete_Failed	OutpostNotReachable	由于与您的 Outposts 主区域的服务链接连接已断开，因此无法删除端点。检查连接，然后重试。
`DeleteEndpoint`	Delete_Failed	InternalError	由于内部错误，无法删除端点。请重试。

有关使用 S3 on Outposts 中的存储桶的更多信息，请参阅使用 S3 on Outposts 存储桶。

下面各部分介绍了如何创建和管理 S3 on Outposts 的端点。

主题

在 Outpost 上创建端点

要将请求路由到 Amazon S3 on Outposts 访问点，您必须创建 S3 on Outposts 端点并进行配置。为了创建端点，您需要使用服务链接建立到 Outposts 主区域的活跃连接。Outposts 上的每个 Virtual Private Cloud (VPC) 都可以有一个关联的端点。有关端点配额的更多信息，请参阅 S3 on Outposts 网络要求。您必须创建一个端点，才能访问 Outposts 桶并执行对象操作。有关更多信息，请参阅端点。

权限

有关创建端点所需权限的更多信息，请参阅 S3 on Outposts 端点的权限。

当您创建端点时，S3 on Outposts 还会在您的 AWS 账户中创建一个服务相关角色。有关更多信息，请参阅将服务相关角色用于 Amazon S3 on Outposts。

以下示例显示如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 创建 S3 on Outposts 端点。

使用 S3 控制台

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts access points（Outposts 访问点）。
3. 选择 Outposts endpoints（Outposts 端点）选项卡。
4. 选择 Create Outposts endpoint（创建 Outposts 端点）。
5. 在 Outpost 下，选择 Outpost 以在其上创建此端点。
6. 在 VPC 下，选择一个还没有端点且也符合 Outposts 端点规则的 VPC。

   通过 Virtual Private Cloud (VPC)，您可以将 AWS 资源启动到您定义的虚拟网络中。这个虚拟网络与您在自己的数据中心中运行的传统网络极其相似，并会为您提供使用的可扩展基础设施的优势 AWS

如果您没有 VPC，请选择 Create VPC（创建 VPC）。有关更多信息，请参阅[创建限制到 Virtual Private Cloud 的接入点]。

7. 选择 Create Outposts endpoint（创建 Outposts 端点）。

使用 AWS CLI

Example

以下 AWS CLI 示例使用 VPC 资源访问类型，为 Outpost 创建端点。VPC 派生自子网。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

以下 AWS CLI 示例使用客户拥有的 IP 地址池（CoIP 池）访问类型为 Outpost 创建端点。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

使用适用于 Java 的 AWS SDK

Example

以下适用于 Java 的 SDK 示例为 Outpost 创建端点。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.CreateEndpointRequest;
import com.amazonaws.services.s3outposts.model.CreateEndpointResult;

public void createEndpoint() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    CreateEndpointRequest createEndpointRequest = new CreateEndpointRequest()
                .withOutpostId("op-0d79779cef3c30a40")
                .withSubnetId("subnet-8c7a57c5")
```

```
                .withSecurityGroupId("sg-ab19e0d1")
                .withAccessType("CustomerOwnedIp")
                .withCustomerOwnedIpv4Pool("ipv4pool-coip-12345678901234567");
    // Use .withAccessType and .withCustomerOwnedIpv4Pool only when the access type is
    // customer-owned IP address pool (CoIP pool)
    CreateEndpointResult createEndpointResult =
 s3OutpostsClient.createEndpoint(createEndpointRequest);
    System.out.println("Endpoint is created and its ARN is " +
 createEndpointResult.getEndpointArn());
}
```

查看 Amazon S3 on Outposts 端点列表

要将请求路由到 Amazon S3 on Outposts 访问点，您必须创建 S3 on Outposts 端点并进行配置。为了创建端点，您需要使用服务链接建立到 Outposts 主区域的活跃连接。Outposts 上的每个 Virtual Private Cloud (VPC) 都可以有一个关联的端点。有关端点配额的更多信息，请参阅 S3 on Outposts 网络要求。您必须创建一个端点，才能访问 Outposts 桶并执行对象操作。有关更多信息，请参阅端点。

以下示例显示了如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 返回 S3 on Outposts 端点的列表。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts access points（Outposts 访问点）。
3. 在 Outposts access points（Outposts 访问点）页上，选择 Outposts endpoints（Outposts 端点）选项卡。
4. 在 Outposts endpoints（Outposts 端点）下，您可以查看 S3 on Outposts 端点的列表。

使用 AWS CLI

以下 AWS CLI 示例列出了与您的账户关联的 AWS Outposts 资源的端点。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 list-endpoints。

```
aws s3outposts list-endpoints
```

使用适用于 Java 的 AWS SDK

以下适用于 Java 的 SDK 示例列出 Outpost 的端点。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 ListEndpoints。

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.ListEndpointsRequest;
import com.amazonaws.services.s3outposts.model.ListEndpointsResult;

public void listEndpoints() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    ListEndpointsRequest listEndpointsRequest = new ListEndpointsRequest();
    ListEndpointsResult listEndpointsResult =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
    System.out.println("List endpoints result is " + listEndpointsResult);
}
```

删除 Amazon S3 on Outposts 终端节点

要将请求路由到 Amazon S3 on Outposts 访问点，您必须创建 S3 on Outposts 端点并进行配置。为了创建端点，您需要使用服务链接建立到 Outposts 主区域的活跃连接。Outposts 上的每个 Virtual Private Cloud (VPC) 都可以有一个关联的端点。有关端点配额的更多信息，请参阅 S3 on Outposts 网络要求。您必须创建一个端点，才能访问 Outposts 桶并执行对象操作。有关更多信息，请参阅端点。

以下示例显示如何使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 删除 S3 on Outposts 终端节点。

使用 S3 控制台

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts access points（Outposts 访问点）。
3. 在 Outposts access points（Outposts 访问点）页上，选择 Outposts endpoints（Outposts 端点）选项卡。
4. 在 Outposts endpoints（Outposts 终端节点）下，选择要删除的终端节点，然后选择 Delete（删除）。

使用 AWS CLI

以下 AWS CLI 示例删除 Outpost 的终端节点。要运行此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3outposts delete-endpoint --endpoint-id example-endpoint-id --outpost-
id op-01ac5d28a6a232904
```

使用适用于 Java 的 AWS 软件开发工具包

以下适用于 Java 的软件开发工具包示例会删除 Outpost 的终端节点。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
import com.amazonaws.arn.Arn;
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.DeleteEndpointRequest;

public void deleteEndpoint(String endpointArnInput) {
    String outpostId = "op-01ac5d28a6a232904";
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    Arn endpointArn = Arn.fromString(endpointArnInput);
    String[] resourceParts = endpointArn.getResource().getResource().split("/");
    String endpointId = resourceParts[resourceParts.length - 1];
    DeleteEndpointRequest deleteEndpointRequest = new DeleteEndpointRequest()
                .withEndpointId(endpointId)
                .withOutpostId(outpostId);
    s3OutpostsClient.deleteEndpoint(deleteEndpointRequest);
    System.out.println("Endpoint with id " + endpointId + " is deleted.");
}
```

使用 S3 on Outposts 对象

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (0UTP0STS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud（VPC）使用接入点和端点连接与 Outposts 桶进行通信。您可以像在 Amazon S3 桶中一样在 Outpost 桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。

对象是 Amazon S3 on Outposts 中存储的基础实体。每个对象都储存在一个存储桶中。您必须使用访问点才能访问 Outpost 存储桶中的任何对象。当您为对象操作指定桶时，可以使用访问点 Amazon 资

源名称（ARN）或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

以下示例显示了 S3 on Outposts 访问点的 ARN 格式，其中包括 Outpost 所属区域的 AWS 区域 代码、AWS 账户 ID、Outpost ID 和访问点名称：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

有关 S3 on Outposts ARN 的更多信息，请参阅S3 on Outposts 的资源 ARN。

对象 ARN 使用以下格式，其中包括 Outpost 归属的 AWS 区域、AWS 账户 ID、Outpost ID、存储桶名称和对象键：

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket1/object/myobject
```

对于 Amazon S3 on Outposts，对象数据始终存储在 Outpost 上。当 AWS 安装 Outpost 机架时，您的数据将保留在 Outpost 的本地，以满足数据驻留要求。您的对象永远不会离开您的 Outpost，也不在 AWS 区域 中。由于 AWS Management Console 托管在区域内，您无法使用控制台上传或管理 Outpost 中的对象。但是，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 和 AWS SDK 通过访问点上传和管理对象。

主题

- 将对象上传到 S3 on Outposts 存储桶
- 使用 AWS SDK for Java 复制 Amazon S3 on Outposts 存储桶中的对象
- 从 Amazon S3 on Outposts 存储桶获取对象
- 列出 Amazon S3 on Outposts 存储桶中的对象
- 删除 Amazon S3 on Outposts 存储桶中的对象
- 使用 HeadBucket 确定是否存在 S3 on Outposts 存储桶以及您是否具有访问权限。
- 使用适用于 Java 的 SDK 执行和管理分段上传
- 使用适用于 S3 on Outposts 的预签名 URL
- Amazon S3 on Outposts 与本地 Amazon EMR on Outposts
- 授权和身份验证缓存

将对象上传到 S3 on Outposts 存储桶

对象是 Amazon S3 on Outposts 中存储的基础实体。每个对象都储存在一个存储桶中。您必须使用访问点才能访问 Outpost 存储桶中的任何对象。当您为对象操作指定桶时，可以使用访问点 Amazon 资源名称（ARN）或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

以下示例显示了 S3 on Outposts 访问点的 ARN 格式，其中包括 Outpost 所属区域的 AWS 区域 代码、AWS 账户 ID、Outpost ID 和访问点名称：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

有关 S3 on Outposts ARN 的更多信息，请参阅S3 on Outposts 的资源 ARN。

对于 Amazon S3 on Outposts，对象数据始终存储在 Outpost 上。当 AWS 安装 Outpost 机架时，您的数据将保留在 Outpost 的本地，以满足数据驻留要求。您的对象永远不会离开您的 Outpost，也不在 AWS 区域 中。由于 AWS Management Console 托管在区域内，您无法使用控制台上传或管理 Outpost 中的对象。但是，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 和 AWS SDK 通过访问点上传和管理对象。

以下 AWS CLI 和AWS SDK for Java 示例演示如何通过使用访问点将对象上传到 S3 on Outposts 存储桶。

AWS CLI

Example

以下示例使用 AWS CLI 将一个名为 `sample-object.xml` 的对象放置到 S3 on Outposts 存储桶 (`s3-outposts:PutObject`)。要使用此命令，请将每个 *`user input placeholder`* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 put-object。

```
aws s3api put-object --bucket arn:aws:s3-
outposts:Region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml --body sample-object.xml
```

SDK for Java

Example

以下示例使用适用于 Java 的 SDK 向 S3 on Outposts 存储桶放置对象。要使用此示例，请将每个 *`user input placeholder`* 替换为您自己的信息。有关更多信息，请参阅 上传对象。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.PutObjectRequest;

import java.io.File;

public class PutObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String stringObjKeyName = "*** String object key name ***";
        String fileObjKeyName = "*** File object key name ***";
        String fileName = "*** Path to file to upload ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Upload a text string as a new object.
            s3Client.putObject(accessPointArn, stringObjKeyName, "Uploaded String
 Object");

            // Upload a file as a new object with ContentType and title specified.
            PutObjectRequest request = new PutObjectRequest(accessPointArn,
 fileObjKeyName, new File(fileName));
            ObjectMetadata metadata = new ObjectMetadata();
            metadata.setContentType("plain/text");
            metadata.addUserMetadata("title", "someTitle");
            request.setMetadata(metadata);
            s3Client.putObject(request);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

使用 AWS SDK for Java 复制 Amazon S3 on Outposts 存储桶中的对象

对象是 Amazon S3 on Outposts 中存储的基础实体。每个对象都储存在一个存储桶中。您必须使用访问点才能访问 Outpost 存储桶中的任何对象。当您为对象操作指定桶时，可以使用访问点 Amazon 资源名称（ARN）或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

以下示例显示了 S3 on Outposts 访问点的 ARN 格式，其中包括 Outpost 所属区域的 AWS 区域 代码、AWS 账户 ID、Outpost ID 和访问点名称：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

有关 S3 on Outposts ARN 的更多信息，请参阅S3 on Outposts 的资源 ARN。

对于 Amazon S3 on Outposts，对象数据始终存储在 Outpost 上。当 AWS 安装 Outpost 机架时，您的数据将保留在 Outpost 的本地，以满足数据驻留要求。您的对象永远不会离开您的 Outpost，也不在 AWS 区域 中。由于 AWS Management Console 托管在区域内，您无法使用控制台上传或管理 Outpost 中的对象。但是，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 和 AWS SDK 通过访问点上传和管理对象。

以下示例显示如何使用 AWS SDK for Java 复制 S3 on Outposts 存储桶中的对象。

使用适用于 Java 的 AWS SDK

以下 S3 on Outposts 示例使用适用于 Java 的 SDK 将一个对象复制到同一个存储桶中的新对象。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;

public class CopyObject {
```

```
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String sourceKey = "*** Source object key ***";
        String destinationKey = "*** Destination object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(accessPointArn,
 sourceKey, accessPointArn, destinationKey);
            s3Client.copyObject(copyObjectRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

从 Amazon S3 on Outposts 存储桶获取对象

对象是 Amazon S3 on Outposts 中存储的基础实体。每个对象都储存在一个存储桶中。您必须使用访问点才能访问 Outpost 存储桶中的任何对象。当您为对象操作指定桶时，可以使用访问点 Amazon 资源名称（ARN）或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

以下示例显示了 S3 on Outposts 访问点的 ARN 格式，其中包括 Outpost 所属区域的 AWS 区域 代码、AWS 账户 ID、Outpost ID 和访问点名称：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

有关 S3 on Outposts ARN 的更多信息，请参阅S3 on Outposts 的资源 ARN。

对于 Amazon S3 on Outposts，对象数据始终存储在 Outpost 上。当 AWS 安装 Outpost 机架时，您的数据将保留在 Outpost 的本地，以满足数据驻留要求。您的对象永远不会离开您的 Outpost，也不在 AWS 区域 中。由于 AWS Management Console 托管在区域内，您无法使用控制台上传或管理 Outpost 中的对象。但是，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 和 AWS SDK 通过访问点上传和管理对象。

下面的示例演示如何使用 AWS Command Line Interface (AWS CLI) 和AWS SDK for Java 下载（获取）对象。

使用 AWS CLI

以下示例使用 AWS CLI 从 S3 on Outposts 存储桶 (`s3-outposts:GetObject`) 获取一个名为 `sample-object.xml` 的对象。要使用此命令，请将每个 *`user input placeholder`* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 get-object。

```
aws s3api get-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point --key testkey sample-object.xml
```

使用适用于 Java 的 AWS SDK

以下 S3 on Outposts 示例使用适用于 Java 的 SDK 获取对象。要使用此示例，请将每个 *`user input placeholder`* 替换为您自己的信息。有关更多信息，请参阅 Amazon Simple Storage Service API 参考中的 GetObject。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ResponseHeaderOverrides;
import com.amazonaws.services.s3.model.S3Object;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class GetObject {
    public static void main(String[] args) throws IOException {
        String accessPointArn = "*** access point ARN ***";
        String key = "*** Object key ***";
```

```
        S3Object fullObject = null, objectPortion = null, headerOverrideObject = null;
        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Get an object and print its contents.
            System.out.println("Downloading an object");
            fullObject = s3Client.getObject(new GetObjectRequest(accessPointArn, key));
            System.out.println("Content-Type: " +
 fullObject.getObjectMetadata().getContentType());
            System.out.println("Content: ");
            displayTextInputStream(fullObject.getObjectContent());

            // Get a range of bytes from an object and print the bytes.
            GetObjectRequest rangeObjectRequest = new GetObjectRequest(accessPointArn,
 key)
                    .withRange(0, 9);
            objectPortion = s3Client.getObject(rangeObjectRequest);
            System.out.println("Printing bytes retrieved.");
            displayTextInputStream(objectPortion.getObjectContent());

            // Get an entire object, overriding the specified response headers, and
 print the object's content.
            ResponseHeaderOverrides headerOverrides = new ResponseHeaderOverrides()
                    .withCacheControl("No-cache")
                    .withContentDisposition("attachment; filename=example.txt");
            GetObjectRequest getObjectRequestHeaderOverride = new
 GetObjectRequest(accessPointArn, key)
                    .withResponseHeaders(headerOverrides);
            headerOverrideObject = s3Client.getObject(getObjectRequestHeaderOverride);
            displayTextInputStream(headerOverrideObject.getObjectContent());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        } finally {
            // To ensure that the network connection doesn't remain open, close any
 open input streams.
            if (fullObject != null) {
                fullObject.close();
            }
            if (objectPortion != null) {
                objectPortion.close();
            }
            if (headerOverrideObject != null) {
                headerOverrideObject.close();
            }
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException {
        // Read the text input stream one line at a time and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line = null;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

列出 Amazon S3 on Outposts 存储桶中的对象

对象是 Amazon S3 on Outposts 中存储的基础实体。每个对象都储存在一个存储桶中。您必须使用访问点才能访问 Outpost 存储桶中的任何对象。当您为对象操作指定桶时，可以使用访问点 Amazon 资源名称（ARN）或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

以下示例显示了 S3 on Outposts 访问点的 ARN 格式，其中包括 Outpost 所属区域的 AWS 区域 代码、AWS 账户 ID、Outpost ID 和访问点名称：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

有关 S3 on Outposts ARN 的更多信息，请参阅S3 on Outposts 的资源 ARN。

> **Note**
>
> 对于 Amazon S3 on Outposts，对象数据始终存储在 Outpost 上。当 AWS 安装 Outpost 机架时，您的数据将保留在 Outpost 的本地，以满足数据驻留要求。您的对象永远不会离开您的 Outpost，也不在 AWS 区域 中。由于 AWS Management Console 托管在区域内，您无法使用控制台上传或管理 Outpost 中的对象。但是，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 和 AWS SDK 通过访问点上传和管理对象。

以下示例显示如何使用 AWS CLI 和AWS SDK for Java 列出 S3 on Outposts 存储桶中的对象。

使用 AWS CLI

以下示例使用 AWS CLI 列出 S3 on Outposts 存储桶 (`s3-outposts:ListObjectsV2`) 中的对象。要使用此命令，请将每个 *`user input placeholder`* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 list-objects-v2。

```
aws s3api list-objects-v2 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

> **Note**
>
> 通过 AWS SDK 将此操作与 Amazon S3 on Outposts 结合使用时，您可以通过以下表单使用 Outposts 访问点 ARN 代替存储桶的名称：`arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-Outposts-Access-Point`。有关 S3 on Outposts ARN 的更多信息，请参阅S3 on Outposts 的资源 ARN。

使用适用于 Java 的 AWS SDK

以下 S3 on Outposts 示例使用适用于 Java 的 SDK 列出桶中的对象。要使用此示例，请将每个 *`user input placeholder`* 替换为您自己的信息。

> **Important**
>
> 此示例使用 ListObjectsV2，这是 ListObjects API 操作的最新版本。我们建议您使用此修订后的 API 操作进行应用程序开发。为了实现向后兼容，Amazon S3 继续支持此 API 操作的先前版本。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Request;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;

public class ListObjectsV2 {

    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            System.out.println("Listing objects");

            // maxKeys is set to 2 to demonstrate the use of
            // ListObjectsV2Result.getNextContinuationToken()
            ListObjectsV2Request req = new
 ListObjectsV2Request().withBucketName(accessPointArn).withMaxKeys(2);
            ListObjectsV2Result result;

            do {
                result = s3Client.listObjectsV2(req);

                for (S3ObjectSummary objectSummary : result.getObjectSummaries()) {
                    System.out.printf(" - %s (size: %d)\n", objectSummary.getKey(),
 objectSummary.getSize());
```

```
                }
                // If there are more than maxKeys keys in the bucket, get a
 continuation token
                // and list the next objects.
                String token = result.getNextContinuationToken();
                System.out.println("Next Continuation Token: " + token);
                req.setContinuationToken(token);
            } while (result.isTruncated());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

删除 Amazon S3 on Outposts 存储桶中的对象

对象是 Amazon S3 on Outposts 中存储的基础实体。每个对象都储存在一个存储桶中。您必须使用访问点才能访问 Outpost 存储桶中的任何对象。当您为对象操作指定桶时，可以使用访问点 Amazon 资源名称（ARN）或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

以下示例显示了 S3 on Outposts 访问点的 ARN 格式，其中包括 Outpost 所属区域的 AWS 区域 代码、AWS 账户 ID、Outpost ID 和访问点名称：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

有关 S3 on Outposts ARN 的更多信息，请参阅S3 on Outposts 的资源 ARN。

对于 Amazon S3 on Outposts，对象数据始终存储在 Outpost 上。当 AWS 安装 Outpost 机架时，您的数据将保留在 Outpost 的本地，以满足数据驻留要求。您的对象永远不会离开您的 Outpost，也不在 AWS 区域 中。由于 AWS Management Console 托管在区域内，您无法使用控制台上传或管理 Outpost 中的对象。但是，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 和 AWS SDK 通过访问点上传和管理对象。

以下示例显示如何使用 AWS Command Line Interface (AWS CLI) 和AWS SDK for Java 从 S3 on Outposts 存储桶删除单个或多个对象。

使用 AWS CLI

以下示例显示如何从 S3 on Outposts 存储桶删除单个或多个对象。

delete-object

以下示例使用 AWS CLI 从 S3 on Outposts 存储桶 (`s3-outposts:DeleteObject`) 删除一个名为 `sample-object.xml` 的对象。要使用此命令，请将每个 *`user input placeholder`* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 delete-object。

```
aws s3api delete-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml
```

delete-objects

以下示例使用 AWS CLI 从 S3 on Outposts 存储桶 (`s3-outposts:DeleteObject`) 删除名为 `sample-object.xml` 和 `test1.text` 的两个对象。要使用此命令，请将每个 *`user input placeholder`* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 delete-objects。

```
aws s3api delete-objects --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --delete file://delete.json

delete.json
{
  "Objects": [
    {
      "Key": "test1.txt"
    },
    {
      "Key": "sample-object.xml"
    }
  ],
  "Quiet": false
```

```
}
```

使用适用于 Java 的 AWS SDK

以下示例显示如何从 S3 on Outposts 存储桶删除单个或多个对象。

DeleteObject

以下 S3 on Outposts 示例使用适用于 Java 的 SDK 删除存储桶中的对象。要使用此示例，请指定 Outpost 的访问点 ARN 和要删除的对象的密钥名称。有关更多信息，请参阅 Amazon Simple Storage Service API 参考中的 DeleteObject。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

public class DeleteObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String keyName = "*** key name ****";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(accessPointArn, keyName));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
```

```
}
```

DeleteObjects

以下 S3 on Outposts 示例使用适用于 Java 的 SDK 向存储桶上传对象，然后删除对象。要使用此示例，请指定 Outpost 的访问点 ARN。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 DeleteObjects。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectsRequest;
import com.amazonaws.services.s3.model.DeleteObjectsRequest.KeyVersion;
import com.amazonaws.services.s3.model.DeleteObjectsResult;

import java.util.ArrayList;

public class DeleteObjects {

    public static void main(String[] args) {
       String accessPointArn = "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point";


        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Upload three sample objects.
            ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
            for (int i = 0; i < 3; i++) {
                String keyName = "delete object example " + i;
                s3Client.putObject(accessPointArn, keyName, "Object number " + i + "
 to be deleted.");
                keys.add(new KeyVersion(keyName));
            }
            System.out.println(keys.size() + " objects successfully created.");
```

```
            // Delete the sample objects.
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest(accessPointArn)
                    .withKeys(keys)
                    .withQuiet(false);

            // Verify that the objects were deleted successfully.
            DeleteObjectsResult delObjRes =
 s3Client.deleteObjects(multiObjectDeleteRequest);
            int successfulDeletes = delObjRes.getDeletedObjects().size();
            System.out.println(successfulDeletes + " objects successfully
 deleted.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用 HeadBucket 确定是否存在 S3 on Outposts 存储桶以及您是否具有访问权限。

对象是 Amazon S3 on Outposts 中存储的基础实体。每个对象都储存在一个存储桶中。您必须使用访问点才能访问 Outpost 存储桶中的任何对象。当您为对象操作指定桶时，可以使用访问点 Amazon 资源名称（ARN）或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

以下示例显示了 S3 on Outposts 访问点的 ARN 格式，其中包括 Outpost 所属区域的 AWS 区域 代码、AWS 账户 ID、Outpost ID 和访问点名称：

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

有关 S3 on Outposts ARN 的更多信息，请参阅S3 on Outposts 的资源 ARN。

> **Note**
>
> 对于 Amazon S3 on Outposts，对象数据始终存储在 Outpost 上。当 AWS 安装 Outpost 机架时，您的数据将保留在 Outpost 的本地，以满足数据驻留要求。您的对象永远不会离开您的 Outpost，也不在 AWS 区域 中。由于 AWS Management Console 托管在区域内，您无法使用控制台上传或管理 Outpost 中的对象。但是，您可以使用 REST API、AWS Command Line Interface (AWS CLI) 和 AWS SDK 通过访问点上传和管理对象。

以下 AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 示例显示如何使用 HeadBucket API 操作确定 Amazon S3 on Outposts 存储桶是否存在以及您是否有权访问该存储桶。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 HeadBucket。

使用 AWS CLI

以下 S3 on Outposts AWS CLI 示例显示如何使用 `head-bucket` 命令确定存储桶是否存在以及您是否有权访问该存储桶。要使用此命令，请将每个 *`user input placeholder`* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 head-bucket。

```
aws s3api head-bucket --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

使用适用于 Java 的 AWS SDK

以下 S3 on Outposts 示例显示如何确定存储桶是否存在以及您是否有权访问该存储桶。要使用此示例，请指定 Outpost 的访问点 ARN。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 HeadBucket。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.HeadBucketRequest;

public class HeadBucket {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
```

```
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.headBucket(new HeadBucketRequest(accessPointArn));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用适用于 Java 的 SDK 执行和管理分段上传

使用 Amazon S3 on Outposts，您可以在 AWS Outposts 资源上创建 S3 存储桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序存储和检索对象。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅 什么是 Amazon S3 on Outposts？

以下示例显示如何结合使用 S3 on Outposts 和 AWS SDK for Java 来执行和管理分段上传。

主题

- 在 S3 on Outposts 存储桶中执行对象的分段上传
- 通过分段上传，在 S3 on Outposts 存储桶中复制大型对象
- 在 S3 on Outposts 存储桶中列出对象的分段
- 检索 S3 on Outposts 存储桶中正在进行的分段上传的列表

在 S3 on Outposts 存储桶中执行对象的分段上传

以下 S3 on Outposts 示例使用适用于 Java 的 SDK 启动、上传和完成对象到存储桶的分段上传。要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。有关更多信息，请参阅使用分段上传操作上传对象。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;

public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
 s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
```

```
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
```

```
        return etags;
    }
```

通过分段上传，在 S3 on Outposts 存储桶中复制大型对象

下面的 S3 on Outposts 示例使用适用于 Java 的 SDK 复制存储桶中的对象。要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。这个例子是通过 使用分段上传复制对象 改写的。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;

public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
 s3Client.getObjectMetadata(metadataRequest);
```

```
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
}
```

在 S3 on Outposts 存储桶中列出对象的分段

以下 S3 on Outposts 示例使用适用于 Java 的 SDK 列出存储桶中对象的分段。要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.List;

public class ListParts {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String keyName = "*** Key name ***";
        String uploadId = "*** Upload ID ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            ListPartsRequest listPartsRequest = new ListPartsRequest(accessPointArn,
 keyName, uploadId);
            PartListing partListing = s3Client.listParts(listPartsRequest);
```

```
            List<PartSummary> partSummaries = partListing.getParts();

            System.out.println(partSummaries.size() + " multipart upload parts");
            for (PartSummary p : partSummaries) {
                System.out.println("Upload part: Part number = \"" + p.getPartNumber()
 + "\", ETag = " + p.getETag());
            }

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

检索 S3 on Outposts 存储桶中正在进行的分段上传的列表

以下 S3 on Outposts 示例说明如何使用适用于 Java 的 SDK 从 Outposts 存储桶检索正在进行的分段上传的列表。要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。这是根据 Amazon S3 列出分段上传 示例改编的一个示例。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListMultipartUploadsRequest;
import com.amazonaws.services.s3.model.MultipartUpload;
import com.amazonaws.services.s3.model.MultipartUploadListing;

import java.util.List;

public class ListMultipartUploads {
    public static void main(String[] args) {
                String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
```

```
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Retrieve a list of all in-progress multipart uploads.
            ListMultipartUploadsRequest allMultipartUploadsRequest = new
 ListMultipartUploadsRequest(accessPointArn);
            MultipartUploadListing multipartUploadListing =
 s3Client.listMultipartUploads(allMultipartUploadsRequest);
            List<MultipartUpload> uploads =
 multipartUploadListing.getMultipartUploads();

            // Display information about all in-progress multipart uploads.
            System.out.println(uploads.size() + " multipart upload(s) in progress.");
            for (MultipartUpload u : uploads) {
                System.out.println("Upload in progress: Key = \"" + u.getKey() + "\",
 id = " + u.getUploadId());
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

使用适用于 S3 on Outposts 的预签名 URL

要授予对 Outpost 本地存储对象的限时访问权限而不更新存储桶策略，您可以使用预签名 URL。借助预签名 URL，作为存储桶的所有者，您可以与您虚拟私有云（VPC）中的个人共享对象，或者向其授予上传或删除对象的权限。

使用 AWS SDK 或 AWS Command Line Interface（AWS CLI）创建预签名 URL 时，您会将该 URL 与某个特定的操作关联。您还可以通过选择自定义到期时间来授予对预签名 URL 的限时访问权限，自定义到期时间最短可为 1 秒，最长可为 7 天。共享预签名 URL 时，VPC 中的个人可以执行嵌入在 URL 中的操作，如同他们就是原始签名用户。URL 在到达其到期时间时将会过期，不再有效。

限制预签名 URL 功能

预签名 URL 的功能受创建它的用户的权限所限制。预签名 URL 实质上是一种不记名令牌，向持有相关 URL 的人授予了访问权限。因此，我们建议您适当地保护它们。

AWS 签名版本 4（SigV4）

使用 AWS 签名版本 4（SigV4）对预签名 URL 请求进行身份认证时要强制执行特定的行为，您可以在存储桶策略和访问点策略中使用条件键。例如，您可以创建一个使用 `s3-outposts:signatureAge` 条件的存储桶策略，以在相关签名的存在时间超过 10 分钟时，拒绝对 `example-outpost-bucket` 存储桶中对象的任何 Amazon S3 on Outposts 预签名 URL 请求。要使用此示例，请将 *`user input placeholders`* 替换为您自己的信息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
        }
    ]
}
```

有关在使用签名版本 4 对预签名 URL 请求进行身份认证时，可用于强制执行特定行为的条件键和其他示例策略的列表，请参阅 AWS 签名版本 4（SigV4）身份认证特定的策略键。

网络路径限制

如果要将预签名 URL 的使用和所有 S3 on Outposts 访问限定为特定的网络路径，您可以编写要求使用特定网络路径的策略。要对发起调用的 IAM 主体设置限制，您可以使用基于身份的 AWS Identity and Access Management（IAM）策略（例如用户、组或角色策略）。要对 S3 on Outposts 资源设置限制，您可以使用基于资源的策略（例如，存储桶和访问点策略）。

对 IAM 主体实施网络路径限制后，要求拥有这些凭证的用户从指定的网络发出请求。对存储桶或访问点实施限制后，要求对该资源的所有请求都必须来自指定的网络。这些限制也适用于预签名 URL 以外的场景。

您使用的 IAM 全局条件取决于端点的类型。如果您使用适用于 S3 on Outposts 的公有端点，请使用 `aws:SourceIp`。如果您使用适用于 S3 on Outposts 的 VPC 端点，请使用 `aws:SourceVpc` 或 `aws:SourceVpce`。

以下 IAM policy 语句要求主体仅从指定的网络范围访问。AWS使用此策略语句时，所有访问都必须来自该范围。这包括有人使用适用于 S3 on Outposts 的预签名 URL 的情况。要使用此示例，请将 *`user input placeholders`* 替换为您自己的信息。

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

有关使用 `aws:SourceIP` AWS 全局条件键将对 S3 on Outposts 存储桶的访问限定为特定网络范围的示例存储桶策略，请参阅使用 S3 on Outposts 设置 IAM。

谁可以创建预签名 URL

具有有效安全凭证的任何人都可以创建预签名 URL。但要使 VPC 中的用户成功地访问对象，必须由拥有执行预签名 URL 所基于的操作权限的人创建该预签名 URL。

您可以使用下面的凭证创建预签名 URL：

- IAM 实例配置文件 – 有效期最长 6 小时。
- AWS Security Token Service – 使用永久凭证（例如，AWS 账户 根用户或 IAM 用户的凭证）签名时，有效期最长 36 小时。
- IAM 用户 – 使用 AWS 签名版本 4 时，有效期最长 7 天。

  要创建有效期最长 7 天的预签名 URL，请首先为所使用的开发工具包委托 IAM 用户凭证（访问密钥和秘密密钥）。然后使用 AWS 签名版本 4 生成预签名 URL。

> **Note**
>
> - 如果您已使用临时令牌创建了预签名 URL，则此 URL 将在令牌过期时过期，即使您使用更晚的到期时间创建了该 URL。
> - 由于预签名 URL 将向持有该 URL 的任何人授予访问 S3 on Outposts 存储桶的权限，我们建议您妥善保护它们。有关保护预签名 URL 的更多信息，请参阅 限制预签名 URL 功能。

S3 on Outposts 何时检查预签名 URL 的到期日期和时间？

在发出 HTTP 请求时，S3 on Outposts 会检查签名 URL 的到期日期和时间。例如，如果客户端刚好在到期时间之前开始下载某个大型文件，即使在下载过程中超过到期时间，下载也会继续进行。但如果连接断开，在客户端试图在超过到期时间后重新开始下载，则下载将会失败。

有关使用预签名 URL 共享或上传对象的更多信息，请参阅以下主题。

主题

- 使用预签名 URL 共享对象
- 生成预签名 URL 以将对象上传到 S3 on Outposts 存储桶

使用预签名 URL 共享对象

要授予对 Outpost 本地存储对象的限时访问权限而不更新存储桶策略，您可以使用预签名 URL。借助预签名 URL，作为存储桶的所有者，您可以与您虚拟私有云（VPC）中的个人共享对象，或者向其授予上传或删除对象的权限。

使用 AWS SDK 或 AWS Command Line Interface（AWS CLI）创建预签名 URL 时，您会将该 URL 与某个特定的操作关联。您还可以通过选择自定义到期时间来授予对预签名 URL 的限时访问权限，自定义到期时间最短可为 1 秒，最长可为 7 天。共享预签名 URL 时，VPC 中的个人可以执行嵌入在 URL 中的操作，如同他们就是原始签名用户。URL 在到达其到期时间时将会过期，不再有效。

创建预签名 URL 时，必须提供您的安全凭证，然后指定以下内容：

- 该 Amazon S3 on Outposts 存储桶的一个访问点 Amazon 资源名称（ARN）
- 一个对象键
- 一个 HTTP方法（GET 用于下载对象）
- 一个到期日期和时间

预签名 URL 仅在指定的有效期内有效。也就是说，您必须在到期日期和时间到达之前启动该 URL 允许的操作。在到期日期和时间到达之前，您可以多次使用预签名 URL。如果您已使用临时令牌创建了预签名 URL，则此 URL 将在令牌过期时过期，即使您使用更晚的到期时间创建了该 URL。

虚拟私有云（VPC）中有权访问预签名 URL 的用户可以访问对象。例如，如果您在桶中具有一段视频，并且桶和对象均为私有，您可以通过生成预签名的 URL 来与其他用户共享视频。由于预签名 URL 将向持有该 URL 的任何人授予访问 S3 on Outposts 存储桶的权限，我们建议您妥善保护这些 URL。有关保护预签名 URL 的更多详细信息，请参阅 限制预签名 URL 功能。

具有有效安全凭证的任何人都可以创建预签名 URL。但必须由拥有执行预签名 URL 所基于的操作权限的人创建该预签名 URL。有关更多信息，请参阅 谁可以创建预签名 URL。

您可以使用 AWS SDK 和 AWS CLI 生成预签名 URL 以共享 S3 on Outposts 存储桶中的对象。有关更多信息，请参阅以下示例。

使用 AWS SDK

您可以使用 AWS SDK 生成可以提供给其他人的预签名 URL，以便他们可以检索对象。

> **Note**
>
> 使用 AWS SDK 生成预签名 URL 时，预签名 URL 的最长到期时间为创建之时起 7 天。

Java

Example

以下示例将生成一个预签名 URL，您可以将其提供给其他人，以便他们可以检索 S3 on Outposts 存储桶的对象。有关更多信息，请参阅 使用适用于 S3 on Outposts 的预签名 URL。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

有关创建和测试有效示例的说明，请参阅《AWS SDK for Java 开发人员指南》中的入门。

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.HttpMethod;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;
```

```
import java.io.IOException;
import java.net.URL;
import java.time.Instant;

public class GeneratePresignedURL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String accessPointArn = "*** access point ARN ***";
        String objectKey = "*** object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Set the presigned URL to expire after one hour.
            java.util.Date expiration = new java.util.Date();
            long expTimeMillis = Instant.now().toEpochMilli();
            expTimeMillis += 1000 * 60 * 60;
            expiration.setTime(expTimeMillis);

            // Generate the presigned URL.
            System.out.println("Generating pre-signed URL.");
            GeneratePresignedUrlRequest generatePresignedUrlRequest =
                    new GeneratePresignedUrlRequest(accessPointArn, objectKey)
                            .withMethod(HttpMethod.GET)
                            .withExpiration(expiration);
            URL url = s3Client.generatePresignedUrl(generatePresignedUrlRequest);

            System.out.println("Pre-Signed URL: " + url.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't
 process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
```

```
}
```

.NET

Example

以下示例将生成一个预签名 URL，您可以将其提供给其他人，以便他们可以检索 S3 on Outposts 存储桶的对象。有关更多信息，请参阅 使用适用于 S3 on Outposts 的预签名 URL。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

有关设置和运行代码示例的信息，请参阅《适用于 .NET 的 AWS SDK 开发人员指南》中的适用于 .NET 的 AWS SDK 入门。

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;

namespace Amazon.DocSamples.S3
{
    class GenPresignedURLTest
    {
        private const string accessPointArn = "*** access point ARN ***";
        private const string objectKey = "*** object key ***";
        // Specify how long the presigned URL lasts, in hours.
        private const double timeoutDuration = 12;
        // Specify your bucket Region (an example Region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            string urlString = GeneratePreSignedURL(timeoutDuration);
        }
        static string GeneratePreSignedURL(double duration)
        {
            string urlString = "";
            try
            {
                GetPreSignedUrlRequest request1 = new GetPreSignedUrlRequest
```

```
                {
                    BucketName = accessPointArn,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration)
                };
                urlString = s3Client.GetPreSignedURL(request1);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            return urlString;
        }
    }
}
```

Python

使用适用于 Python (Boto3) 的 SDK 生成预签名 URL 以共享对象。例如，使用 Boto3 客户端和 `generate_presigned_url` 函数生成一个允许您 `GET` 对象的预签名 URL。

```
import boto3
    url = boto3.client('s3').generate_presigned_url(
    ClientMethod='get_object',
    Params={'Bucket': 'ACCESS_POINT_ARN', 'Key': 'OBJECT_KEY'},
    ExpiresIn=3600)
```

有关使用适用于 Python (Boto3) 的 SDK 生成预签名 URL 的更多信息，请参阅《AWS SDK for Python (Boto) API 参考》中的 Python。

使用 AWS CLI

以下示例 AWS CLI 命令将为一个 S3 on Outposts 存储桶生成一个预签名 URL。要使用此示例，请将 *`user input placeholders`* 替换为您自己的信息。

> **Note**
>
> 使用 AWS CLI 生成预签名 URL 时，预签名 URL 的最长到期时间为创建之时起 7 天。

```
aws s3 presign s3://arn:aws:s3-outposts:us-east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-point/mydoc.txt --expires-in 604800
```

有关更多信息，请参阅《AWS CLI 命令参考》中的 presign。

生成预签名 URL 以将对象上传到 S3 on Outposts 存储桶

要授予对 Outpost 本地存储对象的限时访问权限而不更新存储桶策略，您可以使用预签名 URL。借助预签名 URL，作为存储桶的所有者，您可以与您虚拟私有云（VPC）中的个人共享对象，或者向其授予上传或删除对象的权限。

使用 AWS SDK 或 AWS Command Line Interface（AWS CLI）创建预签名 URL 时，您会将该 URL 与某个特定的操作关联。您还可以通过选择自定义到期时间来授予对预签名 URL 的限时访问权限，自定义到期时间最短可为 1 秒，最长可为 7 天。共享预签名 URL 时，VPC 中的个人可以执行嵌入在 URL 中的操作，如同他们就是原始签名用户。URL 在到达其到期时间时将会过期，不再有效。

创建预签名 URL 时，必须提供您的安全凭证，然后指定以下内容：

- 该 Amazon S3 on Outposts 存储桶的一个访问点 Amazon 资源名称（ARN）
- 一个对象键
- 一个 HTP 方法（PUT 用于上传对象）
- 一个到期日期和时间

预签名 URL 仅在指定的有效期内有效。也就是说，您必须在到期日期和时间到达之前启动该 URL 允许的操作。在到期日期和时间到达之前，您可以多次使用预签名 URL。如果您已使用临时令牌创建预签名 URL，则此 URL 将在令牌过期时过期，即使创建的 URL 的到期时间更晚也是如此。

如果预签名 URL 允许的操作由多个步骤构成（例如分段上传），则必须在到期时间前启动所有步骤。如果 S3 on Outposts 尝试使用某个已过期的 URL 启动某个步骤，您将会遇到错误。

虚拟私有云（VPC）中有权访问预签名 URL 的用户可上传对象。例如，VPC 中有权访问预签名 URL 的用户可以将对象上传到您的存储桶。由于预签名 URL 将向 VPC 中有权访问预签名 URL 的任何用户

授予访问 S3 on Outposts 存储桶的权限，我们建议您妥善保护这些 URL。有关保护预签名 URL 的更多详细信息，请参阅 限制预签名 URL 功能。

具有有效安全凭证的任何人都可以创建预签名 URL。但必须由拥有执行预签名 URL 所基于的操作权限的人创建该预签名 URL。有关更多信息，请参阅谁可以创建预签名 URL。

使用 AWS SDK 为 S3 on Outposts 对象操作生成预签名 URL

Java

SDK for Java 2.x

此示例演示了如何生成一个预签名 URL，以便在限定时间内用于将对象上传到某个 S3 on Outposts 存储桶。有关更多信息，请参阅使用适用于 S3 on Outposts 的预签名 URL。

```
   public static void signBucket(S3Presigner presigner, String
outpostAccessPointArn, String keyName) {

       try {
           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(accessPointArn)
                   .key(keyName)
                   .contentType("text/plain")
                   .build();

           PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))
                   .putObjectRequest(objectRequest)
                   .build();

           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);


           String myURL = presignedRequest.url().toString();
           System.out.println("Presigned URL to upload a file to: " +myURL);
           System.out.println("Which HTTP method must be used when uploading a
file: " +
                   presignedRequest.httpRequest().method());

           // Upload content to the S3 on Outposts bucket by using this URL.
           URL url = presignedRequest.url();
```

```
            // Create the connection and use it to upload the new object by using
the presigned URL.
            HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestProperty("Content-Type","text/plain");
            connection.setRequestMethod("PUT");
            OutputStreamWriter out = new
OutputStreamWriter(connection.getOutputStream());
            out.write("This text was uploaded as an object by using a presigned
URL.");
            out.close();

            connection.getResponseCode();
            System.out.println("HTTP response code is " +
connection.getResponseCode());

        } catch (S3Exception e) {
            e.getStackTrace();
        } catch (IOException e) {
            e.getStackTrace();
        }
    }
```

Python

适用于 Python (Boto3) 的 SDK

此示例演示了如何生成可在限定时间内执行某个 S3 on Outposts 操作的预签名 URL。有关更多信息，请参阅使用适用于 S3 on Outposts 的预签名 URL。要使用该 URL 发出请求，请使用 `Requests` 程序包。

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)
```

```
def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned S3 on Outposts URL that can be used to perform an
 action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds that the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.",
 client_method)
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('-'*88)
    print("Welcome to the Amazon S3 on Outposts presigned URL demo.")
    print('-'*88)

    parser = argparse.ArgumentParser()
    parser.add_argument('accessPointArn', help="The name of the S3 on Outposts
 access point ARN.")
    parser.add_argument(
        'key', help="For a GET operation, the key of the object in S3 on
 Outposts. For a "
                    "PUT operation, the name of a file to upload.")
    parser.add_argument(
        'action', choices=('get', 'put'), help="The action to perform.")
    args = parser.parse_args()
```

```
    s3_client = boto3.client('s3')
    client_action = 'get_object' if args.action == 'get' else 'put_object'
    url = generate_presigned_url(
        s3_client, client_action, {'Bucket': args.accessPointArn, 'Key':
 args.key}, 1000)

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == 'get':
        response = requests.get(url)
    elif args.action == 'put':
        print("Putting data to the URL.")
        try:
            with open(args.key, 'r') as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(f"Couldn't find {args.key}. For a PUT operation, the key must
 be the "
                  f"name of a file that exists on your computer.")

    if response is not None:
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print('-'*88)


if __name__ == '__main__':
    usage_demo()
```

Amazon S3 on Outposts 与本地 Amazon EMR on Outposts

Amazon EMR 是一个托管式集群平台，可简化在 AWS 上运行大数据框架（如 Apache Hadoop 和 Apache Spark）以处理和分析海量数据的操作。通过使用这些框架和相关的开源项目，您可以处理用于分析目的的数据和业务情报工作负载。Amazon EMR 还可以协助您转换大量数据并将数据移入/移出其它 AWS 数据存储和数据库，并支持 Amazon S3 on Outposts。有关更多信息，请参阅《Amazon EMR 管理指南》中的 Amazon EMR on Outposts。

对于 Amazon S3 on Outposts，Amazon EMR 在版本 7.0.0 中开始支持 Apache Hadoop S3A 连接器。早期版本的 Amazon EMR 不支持本地 S3 on Outposts，也不支持 EMR 文件系统（EMRFS）。

受支持的应用程序

Amazon EMR 与 Amazon S3 on Outposts 结合使用可支持以下应用程序：

- Hadoop
- Spark
- Hue
- Hive
- Sqoop
- Pig
- Hudi
- Flink

有关更多信息，请参阅《Amazon EMR 版本指南》。

创建和配置 Amazon S3 on Outposts 桶

Amazon EMR 将 AWS SDK for Java 与 Amazon S3 on Outposts 结合使用来存储输入数据和输出数据。您的 Amazon EMR 日志文件存储在您选择的区域 Amazon S3 位置，而不是本地存储在 Outpost 上。有关更多信息，请参阅《Amazon EMR 管理指南》中的 Amazon EMR 日志。

为了符合 Amazon S3 和 DNS 的要求，S3 on Outposts 桶具有命名限制和局限性。有关更多信息，请参阅创建 S3 on Outposts 存储桶。

在 Amazon EMR 版本 7.0.0 及更高版本中，您可以将 Amazon EMR 与 S3 on Outposts 和 S3A 文件系统结合使用。

先决条件

S3 on Outposts 权限 – 当您创建 Amazon EMR 实例配置文件时，您的角色必须包含 S3 on Outposts 的 AWS Identity and Access Management（IAM）命名空间。S3 on Outposts 具有自己的命名空间 `s3-outposts*`。有关使用此命名空间的示例策略，请参阅使用 S3 on Outposts 设置 IAM。

S3A 连接器 – 要将您的 EMR 集群配置为访问 Amazon S3 on Outposts 桶中的数据，您必须使用 Apache Hadoop S3A 连接器。要使用该连接器，请确保所有 S3 URI 都使用 `s3a` 方案。如果不是这样，您可以配置用于 EMR 集群的文件系统实现，以便 S3 URI 可以与 S3A 连接器结合使用。

要将文件系统实现配置为与 S3A 连接器结合使用，请为 EMR 集群使用 `fs.file_scheme.impl` 和 `fs.AbstractFileSystem.file_scheme.impl` 配置属性，其中 *file_scheme* 对应于您拥有的 S3 URI 类型。要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。例如，要更改使用 `s3` 方案的 S3 URI 的文件系统实现，请指定以下集群配置属性：

```
[
  {
"Classification": "core-site",
    "Properties": {
    "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
    "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

要使用 S3A，请将 `fs.file_scheme.impl` 配置属性设置为 `org.apache.hadoop.fs.s3a.S3AFileSystem`，并将 `fs.AbstractFileSystem.file_scheme.impl` 属性设置为 `org.apache.hadoop.fs.s3a.S3A`。

例如，如果您要访问路径 `s3a://bucket/...`，请将 `fs.s3a.impl` 属性设置为 `org.apache.hadoop.fs.s3a.S3AFileSystem`，然后将 `fs.AbstractFileSystem.s3a.impl` 属性设置为 `org.apache.hadoop.fs.s3a.S3A`。

开始将 Amazon EMR 与 Amazon S3 on Outposts 结合使用

以下主题介绍了如何开始将 Amazon EMR 与 Amazon S3 on Outposts 结合使用。

主题

创建权限策略

在创建使用 Amazon S3 on Outposts 的 EMR 集群之前，您必须创建一个 IAM policy 以附加到该集群的 Amazon EC2 实例配置文件。该策略必须具有访问 S3 on Outposts 接入点 Amazon 资源名称

（ARN）的权限。有关为 S3 on Outposts 创建 IAM policy 的更多信息，请参阅使用 S3 on Outposts 设置 IAM。

以下示例策略显示如何授予所需的权限。创建策略后，将策略附加到您用于创建 EMR 集群的实例配置文件角色，如 the section called “创建和配置集群” 部分所述。要使用此示例，请将 *user input placeholders* 替换为您自己的信息。

```
{
"Version":"2012-10-17",
  "Statement": [
        {
  "Effect": "Allow",
            "Resource": "arn:aws:s3-outposts:us-
west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name,
            "Action": [
                "s3-outposts:*"
            ]
        }
    ]

 }
```

创建和配置集群

要创建在 S3 on Outposts 中运行 Spark 的集群，请在控制台中完成以下步骤。

创建在 S3 on Outposts 中运行 Spark 的集群

1. 通过以下链接打开 Amazon EMR 控制台：https://console.aws.amazon.com/elasticmapreduce/。
2. 在左侧导航窗格中，选择集群。
3. 选择创建集群。
4. 对于 Amazon EMR 版本，请选择 emr-7.0.0 或更高版本。
5. 对于应用程序捆绑包，请选择 Spark 交互式。然后，选择要包含在集群中的任何其它受支持的应用程序。
6. 要启用 Amazon S3 on Outposts，请输入您的配置设置。

   示例配置设置

   要使用以下示例配置设置，请将 *user input placeholders* 替换为您自己的信息。

```
[
 {
   "Classification": "core-site",
   "Properties": {
     "fs.s3a.bucket.DOC-EXAMPLE-BUCKET.accesspoint.arn": "arn:aws:s3-outposts:us-
west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name"
     "fs.s3a.committer.name": "magic",
     "fs.s3a.select.enabled": "false"
    }
  },
  {
    "Classification": "hadoop-env",
    "Configurations": [
      {
        "Classification": "export",
        "Properties": {
          "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
          }
       }
     ],
     "Properties": {}
   },
   {
     "Classification": "spark-env",
     "Configurations": [
       {
         "Classification": "export",
         "Properties": {
           "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
         }
       }
      ],
      "Properties": {}
     },
     {
      "Classification": "spark-defaults",
      "Properties": {
        "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-11-amazon-
corretto.x86_64",
        "spark.sql.sources.fastS3PartitionDiscovery.enabled": "false"
      }
     }
  ]
```

7. 在联网部分中，选择您的 AWS Outposts 机架上的虚拟私有云（VPC）和子网。有关 Amazon EMR on Outposts 的更多信息，请参阅《Amazon EMR 管理指南》中的 AWS Outposts 上的 EMR 集群。
8. 在 Amazon EMR 的 EC2 实例配置文件部分，选择附加了您之前创建的权限策略的 IAM 角色。
9. 配置剩余的集群设置，然后选择创建集群。

配置概述

下表描述了 S3A 和 Spark 配置，以及在设置将 S3 on Outposts 与 Amazon EMR 结合使用的集群时要为这些配置的参数指定的值。

S3A 配置

参数	默认值	S3 on Outposts 的必需值	说明
`fs.s3a.aws.credentials.provider`	如果未指定，S3A 将在区域桶中查找具有 Outposts 桶名称的 S3。	S3 on Outposts 桶的接入点 ARN	Amazon S3 on Outposts 支持将纯 Virtual Private Cloud (VPC) 接入点作为访问 Outposts 存储桶的唯一方式。
`fs.s3a.committer.name`	`file`	`magic`	Magic 提交程序是 S3 on Outposts 唯一支持的提交程序。
`fs.s3a.select.enabled`	`TRUE`	`FALSE`	Outposts 上不支持 S3 Select。
`JAVA_HOME`	`/usr/lib/jvm/java-8`	`/usr/lib/jvm/java-11-amazon-corretto.x86_64`	S3A 上的 S3 on Outposts 需要 Java 版本 11。

Spark 配置

参数	默认值	S3 on Outposts 的必需值	说明
`spark.sql.sources.fastS3PartitionDiscovery.enabled`	`TRUE`	`FALSE`	S3 on Outposts 不支持快速分区。
`spark.executorEnv.JAVA_HOME`	`/usr/lib/jvm/java-8`	`/usr/lib/jvm/java-11-amazon-corretto.x86_64`	S3A 上的 S3 on Outposts 需要 Java 版本 11。

注意事项

当您将 Amazon EMR 与 S3 on Outposts 桶集成时，请考虑以下几点：

- Amazon EMR 版本 7.0.0 及更高版本支持 Amazon S3 on Outposts。
- 将 S3 on Outposts 与 Amazon EMR 结合使用时需要 S3A 连接器。只有 S3A 具有与 S3 on Outposts 桶交互所需的功能。有关 S3A 连接器设置的信息，请参阅先决条件。
- Amazon S3 on Outposts 对于 Amazon EMR 仅支持采用 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。有关更多信息，请参阅the section called “数据加密”。
- Amazon S3 on Outposts 不支持使用 S3A FileOutputCommitter 进行写入。在 S3 on Outposts 桶上使用 S3A FileOutputCommitter 进行写入会导致以下错误：InvalidStorageClass：您指定的存储类无效。
- Amazon EMR Serverless 或 Amazon EMR on EKS 不支持 Amazon S3 on Outposts。
- Amazon EMR 日志存储在您选择的区域 Amazon S3 位置，而不是本地存储在 S3 on Outposts 桶中。

授权和身份验证缓存

S3 on Outposts 可在 Outposts 机架上安全地在本地缓存身份验证和授权数据。该缓存会针对每个请求，删除到父 AWS 区域的往返行程。这能够消除网络往返带来的可变性。借助 S3 on Outposts 中的身份验证和授权缓存功能，您可以获得与 Outposts 和AWS 区域 之间的连接延迟无关的一致延迟。

当您发出 S3 on Outposts API 请求时，身份验证和授权数据会被安全地缓存。然后，缓存的数据可用于对后续 S3 对象 API 请求进行身份验证。在使用签名版本 4A（SigV4A）对请求进行签名时，S3 on Outposts 仅缓存身份验证和授权数据。缓存本地存储在 S3 on Outposts 服务中的 Outposts 上。当您发出 S3 API 请求时，它会异步刷新。缓存已加密，Outposts 上不会存储任何纯文本加密密钥。

当 Outpost 连接到 AWS 区域时，缓存的有效期最长为 10 分钟。当您发出 S3 on Outposts API 请求时，它会异步刷新，以确保使用最新策略。如果 Outpost 与 AWS 区域断开连接，则缓存的有效期最长为 12 小时。

配置授权和身份验证缓存

S3 on Outposts 会针对使用 Sigv4a 算法进行签名的请求，自动缓存身份验证和授权数据。有关更多信息，请参阅《AWS Identity and Access Management 用户指南》中的对 AWS API 请求进行签名。在最新版本的 AWS SDK 中提供了 SigV4A 算法。您可以通过依赖 AWS 公共运行时（CRT）库来获取它。

您需要使用最新版本的 AWS SDK 并安装最新版本的 CRT。例如，您可以通过运行 `pip install awscrt` 使用 Boto3 来获取最新版本的 CRT。

S3 on Outposts 不会针对使用 SigV4 算法进行签名的请求，缓存身份验证和授权数据。

验证 SigV4A 签名

您可以使用 AWS CloudTrail 来验证是否已使用 SigV4A 对请求进行了签名。有关为 S3 on Outposts 设置 CloudTrail 的更多信息，请参阅使用 AWS CloudTrail 日志监控 S3 on Outposts。

配置 CloudTrail 后，您可以在 CloudTrail 日志的 `SignatureVersion` 字段中验证是否已对请求进行了签名。使用 SigV4A 进行了签名的请求的 `SignatureVersion` 将设置为 `AWS4-ECDSA-P256-SHA256`。使用 SigV4 进行了签名的请求的 `SignatureVersion` 将设置为 `AWS4-HMAC-SHA256`。

S3 on Outposts 中的安全性

AWS 十分重视云安全性。为了满足对安全性最敏感的组织的需求，我们打造了具有超高安全性的数据中心和网络架构。作为 AWS 客户，您也将从这些数据中心和网络架构受益。

安全性是 AWS 和您的共同责任。shared responsibility model（责任共担模式）（责任共担模式）（责任共担模式）将其描述为云的安全性和云中的安全性：

- 云的安全性 – AWS 负责保护在 AWS Cloud 中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。第三方审核员定期测试和验证我们的安全性的有效性，作为 AWS 合规性计划的一部分。要了解适用于 Amazon S3 on Outposts 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性 – 您的责任由您使用的 AWS 服务 决定。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 S3 on Outposts 时应用责任共担模式。以下主题显示了如何配置 S3 on Outposts 以实现您的安全性和合规性目标。您还会了解如何使用其他 AWS 服务以帮助您监控和保护 S3 on Outposts 资源。

主题

S3 on Outposts 中的数据加密

原定设置情况下，存储在 Amazon S3 on Outposts 中的所有数据都使用带 Amazon S3 托管式加密密钥 (SSE-S3) 的服务器端加密进行加密。有关更多信息，请参阅使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）。

您可以选择使用带客户提供的加密密钥的服务器端加密 (SSE-C)。要使用 SSE-C，请在对象 API 请求中指定加密密钥。服务器端加密仅加密对象数据而非加密对象元数据。有关更多信息，请参阅使用具有客户提供的密钥的服务器端加密（SSE-C）。

> **Note**
>
> S3 on Outposts 不支持带 AWS Key Management Service (AWS KMS) 密钥的服务器端加密 (SSE-KMS)。

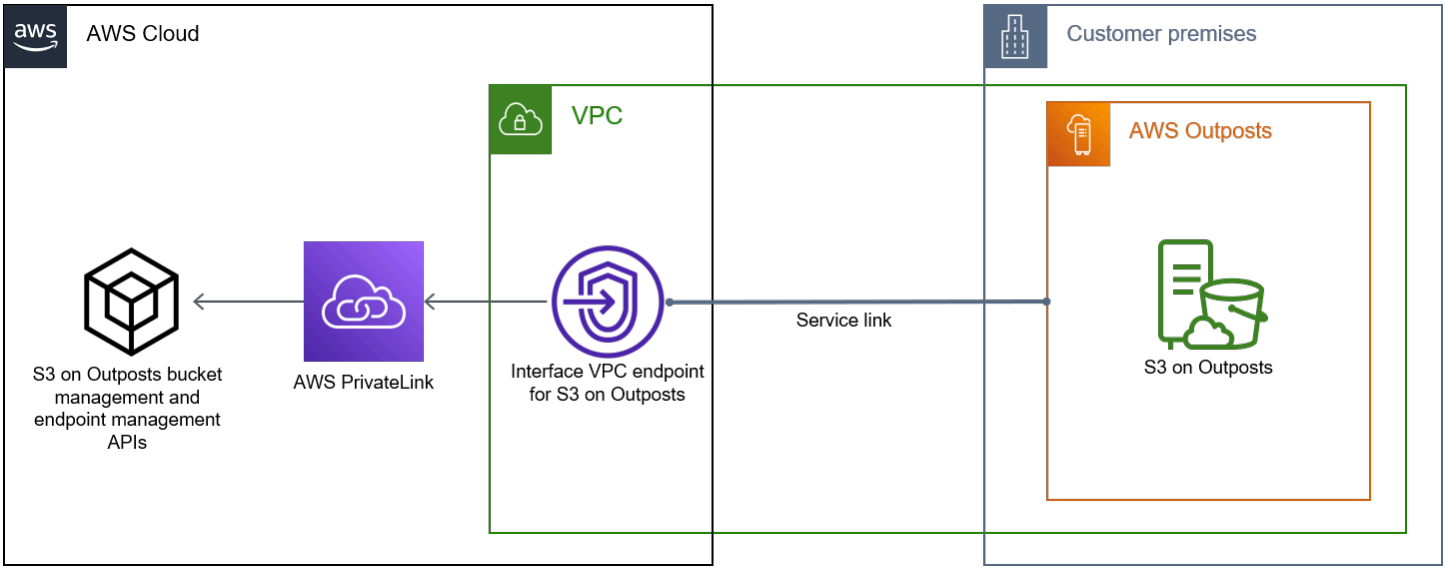
S3 on Outposts 的 AWS PrivateLink

S3 on Outposts 支持 AWS PrivateLink，后者通过虚拟专用网络中的私有端点，提供对 S3 on Outposts 存储的直接管理访问。这样，您就可以使用虚拟私有云（VPC）中的私有 IP 地址，简化内部网络架构并对 Outposts 对象存储执行管理操作。使用 AWS PrivateLink，无需使用公有 IP 地址或代理服务器。

通过将 AWS PrivateLink 用于 Amazon S3 on Outposts，您可以在虚拟私有云（VPC）中预调配接口 VPC 端点，以访问 S3 on Outposts 桶管理和端点管理 API。通过虚拟专用网络（VPN）或 AWS Direct Connect，可以直接从部署在 VPC 中或本地的应用程序访问接口 VPC 端点。您可以通过 AWS PrivateLink 访问桶和端点管理 API。AWS PrivateLink 不支持数据传输 API 操作，例如 GET、PUT 和类似的 API。这些操作已经通过 S3 on Outposts 端点和接入点配置私密传输。有关更多信息，请参阅 S3 on Outposts 的网络。

接口端点由一个或多个弹性网络接口 (ENI) 代表，这些接口是从 VPC 中的子网分配的私有 IP 地址。向 S3 on Outposts 的接口端点发出的请求将自动路由到 AWS 网络上的 S3 on Outposts 桶和端点管理 API。您还可以通过 AWS Direct Connect 或 AWS Virtual Private Network (AWS VPN) 从本地部署应用程序访问 VPC 中的接口端点。有关如何将 VPC 与本地网络连接的更多信息，请参阅 AWS Direct Connect 用户指南和 AWS Site-to-Site VPN 用户指南。

接口端点通过 AWS 网络和通过 AWS PrivateLink 路由对 S3 on Outposts 桶和端点管理 API 的请求，如下图所示。



有关接口端点的一般信息，请参阅 AWS PrivateLink 指南中的接口 VPC 端点 (AWS PrivateLink)。

主题

- 限制和局限性
- 访问 S3 on Outposts 接口端点
- 更新本地 DNS 配置
- 为 S3 on Outposts 创建 VPC 端点
- 为 S3 on Outposts 创建桶策略和 VPC 端点策略

限制和局限性

当您通过 AWS PrivateLink 访问 S3 on Outposts 桶和端点管理 API 时，将适用 VPC 限制。有关更多信息，请参阅 AWS PrivateLink 指南中的接口端点属性和限制以及 AWS PrivateLink 配额。

此外，AWS PrivateLink 不支持以下内容：

- 美国联邦信息处理标准 (FIPS) 端点
- S3 on Outposts 数据传输 API，例如，GET、PUT 和类似的对象 API 操作。
- 私有 DNS

访问 S3 on Outposts 接口端点

要使用 AWS PrivateLink 访问 S3 on Outposts 桶和端点管理 API，您必须更新应用程序以使用特定于端点的 DNS 名称。创建接口端点时，AWS PrivateLink 会生成两种特定于端点的 S3 on Outposts 名称：区域 和可用区。

- 区域 DNS 名称 – 包括唯一的 VPC 端点 ID、服务标识符、AWS 区域和 `vpce.amazonaws.com`，例如 *`vpce-1a2b3c4d-5e6f`*`.s3-outposts.us-east-1.vpce.amazonaws.com`。
- 可用区 DNS 名称 – 包括唯一的 VPC 端点 ID、可用区、服务标识符、AWS 区域和 `vpce.amazonaws.com`，例如 *`vpce-1a2b3c4d-5e6f`*`-us-east-1a.s3-outposts.us-east-1.vpce.amazonaws.com`。如果您的架构隔离了可用区，则可以使用此选项。例如，您可以将可用区 DNS 名称用于故障控制或降低区域数据传输成本。

> **Important**
>
> S3 on Outposts 接口端点是从公有 DNS 域解析出来的。S3 on Outposts 不支持私有 DNS。将 `--endpoint-url` 参数用于所有桶和端点管理 API。

AWS CLI 示例

使用 `--region` 和 `--endpoint-url` 参数，通过 S3 on Outposts 接口端点访问桶管理和端点管理 API。

Example ：使用端点 URL 列出具有 S3 控制 API 的桶

在以下示例中，将区域 *us-east-1*、VPC 端点 URL *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* 和账户 ID *111122223333* 替换为相应的信息。

```
aws s3control list-regional-buckets --region us-east-1 --endpoint-url
 https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com --account-
id 111122223333
```

AWS SDK 示例

将 SDK 更新到最新版本，然后将客户端配置为使用端点 URL 访问 S3 on Outposts 接口端点的 S3 控制 API。有关更多信息，请参阅适用于 AWS PrivateLink 的 AWS SDK 示例。

SDK for Python (Boto3)

Example ：使用端点 URL 访问 S3 控制 API

在以下示例中，将区域 *us-east-1* 和 VPC 端点 URL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* 替换为相应的信息。

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com'
)
```

有关更多信息，请参阅《Boto3 开发人员指南》中的适用于 Amazon S3 的 AWS PrivateLink。

SDK for Java 2.x

Example ：使用端点 URL 访问 S3 控制 API

在以下示例中，将 VPC 端点 URL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* 和区域 *Region.US_EAST_1* 替换为相应的信息。

```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://vpce-1a2b3c4d-5e6f.s3-outposts.us-
east-1.vpce.amazonaws.com"))
                                  .build()
```

有关更多信息，请参阅《AWS SDK for Java API 参考》中的 S3ControlClient。

更新本地 DNS 配置

使用特定于端点的 DNS 名称访问适用于 S3 on Outposts 桶管理和端点管理 API 的接口端点时，您不必更新本地 DNS 解析程序。您可以使用来自公有 S3 on Outposts DNS 域的接口端点的私有 IP 地址解析特定于端点的 DNS 名称。

为 S3 on Outposts 创建 VPC 端点

要为 S3 on Outposts 创建 VPC 接口端点，请参阅《AWS PrivateLink 指南》中的创建 VPC 端点。

为 S3 on Outposts 创建桶策略和 VPC 端点策略

您可以向 VPC 端点附加用于控制对 S3 on Outposts 的访问的端点策略。您还可以在 S3 on Outposts 桶策略中使用 `aws:sourceVpce` 条件来限制从特定 VPC 端点访问特定的桶。使用 VPC 端点策略，您可以控制对 S3 on Outposts 桶管理 API 和端点管理 API 的访问。使用桶策略，您可以控制对 S3 on Outposts 桶管理 API 的访问。但是，您无法使用 `aws:sourceVpce` 管理对 S3 on Outposts 的对象操作的访问。

S3 on Outposts 的访问策略指定以下信息：

- 允许或拒绝其操作的 AWS Identity and Access Management (IAM) 主体。
- 允许或拒绝的 S3 控制操作。
- 允许或拒绝其操作的 S3 on Outposts 资源。

以下示例显示了限制对桶或端点的访问的策略。有关 VPC 连接的更多信息，请参阅 AWS 白皮书 Amazon Virtual Private Cloud 连接性选项中的 Network-to-VPC connectivity options（从网络到 VPC 的连接性选项）。

Important

- 当您如本节所述对 VPC 端点应用示例策略时，您可能会无意中阻止对桶的访问权限。桶权限会限制桶访问源自 VPC 端点的连接，而这可能会阻止到桶的所有连接。有关如何修复此问题的信息，请参阅我的桶策略有错误的 VPC 或 VPC 端点 ID。AWS Support 知识中心内的如何修复策略才能访问桶？。
- 在使用以下示例桶策略之前，请将 VPC 端点 ID 替换为适合您的使用案例的值。否则，您将无法访问您的桶。
- 如果您的策略仅允许从特定 VPC 端点访问 S3 on Outposts 桶，它将禁用通过控制台访问该桶，因为控制台请求不是源自指定的 VPC 端点。

主题

示例：限制从 VPC 端点对特定桶的访问

您可以创建一个端点策略以仅允许访问特定的 S3 on Outposts 桶。以下策略将 GetBucketPolicy 操作的访问权限仅限于 *example-outpost-bucket*。要使用此策略，请将示例值替换为您自己的值。

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
      "Principal": {"AWS":"111122223333"},
      "Action": "s3-outposts:GetBucketPolicy",
      "Effect": "Allow",
      "Resource": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-
bucket"
    }
  ]
}
```

示例：在 S3 on Outposts 桶策略中拒绝从特定 VPC 端点访问

以下 S3 on Outposts 桶策略拒绝通过 *vpce-1a2b3c4d* VPC 端点访问 *example-outpost-bucket* 桶上的 GetBucketPolicy。

`aws:sourceVpce` 条件指定端点，不需要 VPC 端点资源的 Amazon 资源名称 (ARN)，只需要端点 ID。要使用此策略，请将示例值替换为您自己的值。

```
{
    "Version": "2012-10-17",
    "Id": "Policy1415115909152",
    "Statement": [
        {
            "Sid": "Deny-access-to-specific-VPCE",
            "Principal": {"AWS":"111122223333"},
            "Action": "s3-outposts:GetBucketPolicy",
            "Effect": "Deny",
            "Resource": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-
bucket",
            "Condition": {
                "StringEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}
            }
        }
    ]
}
```

AWS 签名版本 4（SigV4）身份认证特定的策略键

下表显示了与 AWS 签名版本 4（SigV4）身份认证有关的条件键，您可以将这些条件键用于 Amazon S3 on Outposts。您可以在存储桶策略中添加这些条件，以便在使用签名版本 4 对请求进行身份认证时强制执行特定的行为。有关示例策略，请参阅 使用与签名版本 4 有关的条件键的存储桶策略示例。有关使用签名版本 4 对请求进行身份认证的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的请求的身份认证（AWS 签名版本 4）。

适用于 **s3-outposts:*** 操作或任何 S3 on Outposts 操作的键

适用的键	描述
s3-outposts:authType	S3 on Outposts 支持多种身份认证方法。要限定传入的请求使用特定的身份认证方法，您可以使用此可选条件键。例如，您可以使用此条件键以仅允许在请求身份认证中使用 HTTP Authorization 标头。 有效值： REST-HEADER REST-QUERY-STRING
s3-outposts:signatureAge	签名在已通过身份认证的请求中的有效期（以毫秒为单位）。 此条件仅适用于预签名 URL。 在签名版本 4 中，签名键的有效期最长为 7 天。因此，签名的有效期最初也为 7 天。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 签名请求简介。您可以使用此条件进一步限制签名有效期。 示例值: 600000
s3-outposts:x-amz-content-sha256	您可以使用此条件键禁止存储桶中未签名的内容。 使用签名版本 4 时，对于使用 Authorization 标头的请求，您需要在签名计算中添加 x-amz-content-sha256 标头，然后将它的值设置为哈希负载。 您可以在存储桶策略中使用此条件键来拒绝上传任何未签名的有效负载。例如： • 拒绝使用 Authorization 标头对请求进行身份认证，但未对有效负载进行签名的上传。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的在单个区块中传输有效负载。 • 拒绝使用预签名 URL 的上传。预签名 URL 始终有一个 UNSIGNED_PAYLOAD 。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的对请求进行身份认证和身份认证方法。

适用的键	描述
	有效值：UNSIGNED-PAYLOAD

使用与签名版本 4 有关的条件键的存储桶策略示例

要使用以下示例，请将 *user input placeholders* 替换为您自己的信息。

Example : **s3-outposts:signatureAge**

以下存储桶策略将在签名的存在时间超过 10 分钟时拒绝对 example-outpost-bucket 中对象的 S3 on Outposts 预签名 URL 请求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
        }
    ]
}
```

Example : **s3-outposts:authType**

以下存储桶策略仅允许使用 Authorization 标头进行请求身份认证的请求。任何预签名的 URL 请求都将被拒绝，因为预签名 URL 使用查询参数来提供请求和身份认证信息。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的身份认证方法。

```
{
```

```
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Allow only requests that use the Authorization header for
 request authentication. Deny presigned URL requests.",
               "Effect": "Deny",
               "Principal": {"AWS":"111122223333"},
               "Action": "s3-outposts:*",
               "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
               "Condition": {
                     "StringNotEquals": {
                           "s3-outposts:authType": "REST-HEADER"
                     }
               }
         }
   ]
}
```

Example : **s3-outposts:x-amz-content-sha256**

以下存储桶策略拒绝任何带有未签名有效负载的上传，例如使用预签名 URL 的上传。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的对请求进行身份认证和身份认证方法。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
               "Sid": "Deny uploads with unsigned payloads.",
               "Effect": "Deny",
               "Principal": {"AWS":"111122223333"},
               "Action": "s3-outposts:*",
               "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/object/
*",
               "Condition": {
                     "StringEquals": {
                           "s3-outposts:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
                     }
               }
         }
   ]
```

```
}
```

适用于 Amazon S3 on Outposts 的 AWS 托管式策略

AWS 托管式策略是由 AWS 创建和管理的独立策略。AWS 托管式策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住，AWS 托管式策略可能不会为您的特定使用场景授予最低权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新在 AWS 托管式策略中定义的权限，则更新会影响该策略所附加到的所有主体身份（用户、组和角色）。当新的 AWS 服务启动或新的 API 操作可用于现有服务时，AWS 最有可能更新 AWS 托管式策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管式策略：AWSS3OnOutpostsServiceRolePolicy

作为服务相关角色 `AWSServiceRoleForS3OnOutposts` 的一部分，帮助您管理网络资源。

要查看此策略的权限，请参阅 AWSS3OnOutpostsServiceRolePolicy。

AWS 托管式策略的 S3 on Outposts 更新

查看自 S3 on Outposts 开始跟踪更改以来，有关此服务的 AWS 托管式策略的更新的详细信息。

更改	说明	日期
S3 on Outposts 添加了 `AWSS3OnOutpostsServiceRolePolicy`	S3 on Outposts 添加了 `AWSS3OnOutpostsServiceRolePolicy` 作为服务相关角色 `AWSServiceRoleForS3OnOutposts` 的一部分，它可帮助您管理网络资源。	2023 年 10 月 3 日
S3 on Outposts 开始跟踪更改	S3 on Outposts 开始跟踪其 AWS 托管式策略的更改。	2023 年 10 月 3 日

将服务相关角色用于 Amazon S3 on Outposts

Amazon S3 on Outposts 使用 AWS Identity and Access Management（IAM）服务相关角色。服务相关角色是一种与 S3 on Outposts 直接关联的独特类型的 IAM 角色。服务相关角色由 S3 on Outposts 预定义，具有服务代表您调用其它 AWS 服务所需的所有权限。

服务相关角色可让您更轻松地设置 S3 on Outposts，因为您不必手动添加必要的权限。S3 on Outposts 定义其服务相关角色的权限，除非另外定义，否则只有 S3 on Outposts 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

只有在首先删除相关资源后，才能删除服务相关角色。这将保护您的 S3 on Outposts 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的 AWS 服务，并查找 Service-linked roles（服务相关角色）列中显示为 Yes（是）的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。

S3 on Outposts 的服务相关角色权限

S3 on Outposts 使用名为 AWSServiceRoleForS3OnOutposts 的服务相关角色来帮助您管理网络资源。

`AWSServiceRoleForS3OnOutposts` 服务相关角色信任以下服务代入该角色：

- `s3-outposts.amazonaws.com`

名为 `AWSS3OnOutpostsServiceRolePolicy` 的角色权限策略允许 S3 on Outposts 对指定资源完成以下操作：

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeCoipPools",
                "ec2:GetCoipPoolUsage",
                "ec2:DescribeAddresses",
```

```
                "ec2:DescribeLocalGatewayRouteTableVpcAssociations"
            ],
            "Resource": "*",
            "Sid": "DescribeVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ],
            "Sid": "CreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForCreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:ipv4pool-ec2/*"
            ],
            "Sid": "AllocateIpAddress"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:AllocateAddress"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:elastic-ip/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForAllocateIpAddress"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:AssociateAddress"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "ReleaseVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": [
                        "CreateNetworkInterface",
                        "AllocateAddress"
                    ],
                    "aws:RequestTag/CreatedBy": [
```

```
                        "S3 On Outposts"
                    ]
                }
            },
            "Sid": "CreateTags"
        }
    ]
}
```

必须配置权限，以允许 IAM 实体（如角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅 IAM 用户指南中的服务相关角色权限。

为 S3 on Outposts 创建服务相关角色

无需手动创建服务相关角色。当您在 AWS Management Console、AWS CLI 或 AWS API 中创建 S3 on Outposts 端点时，S3 on Outposts 会为您创建服务相关角色。

如果删除此服务相关角色，然后需要再次创建，可以使用相同流程在账户中重新创建此角色。当您创建 S3 on Outposts 端点时，S3 on Outposts 会再次为您创建服务相关角色。

您也可以使用 IAM 控制台为 S3 on Outposts 使用案例创建服务相关角色。在 AWS CLI 或 AWS API 中，使用 `s3-outposts.amazonaws.com` 服务名称创建服务相关角色。有关更多信息，请参阅《IAM 用户指南》 中的创建服务相关角色。如果您删除了此服务相关角色，则可以使用此相同过程再次创建角色。

编辑 S3 on Outposts 的服务相关角色

S3 on Outposts 不允许您编辑 `AWSServiceRoleForS3OnOutposts` 服务相关角色。这包括角色的名称，因为可能有不同的实体引用该角色。但是可以使用 IAM 编辑角色说明。有关更多信息，请参阅 IAM 用户指南中的编辑服务相关角色。

删除 S3 on Outposts 的服务相关角色

如果不再需要使用某个需要服务相关角色的特征或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除服务相关角色的资源，然后才能手动删除它。

> **Note**
>
> 如果在您试图删除资源时 S3 on Outposts 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

删除 AWSServiceRoleForS3OnOutposts 角色使用的 S3 on Outposts 资源

1. 在您的 AWS 账户中跨所有 AWS 区域删除 S3 on Outposts 端点。
2. 使用 IAM 删除服务相关角色。

   使用 IAM 控制台，即 AWS CLI 或 AWS API 来删除 `AWSServiceRoleForS3OnOutposts` 服务相关角色。有关更多信息，请参阅 IAM 用户指南中的删除服务相关角色。

S3 on Outposts 服务相关角色的受支持区域

S3 on Outposts 支持在提供服务相关角色的所有 AWS 区域中使用该服务。有关更多信息，请参阅 S3 on Outposts 区域和端点。

管理 S3 on Outposts 存储

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (OUTPOSTS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud（VPC）使用接入点和端点连接与 Outposts 桶进行通信。您可以像在 Amazon S3 桶中一样在 Outpost 桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅什么是 Amazon S3 on Outposts？。

有关管理共享 Amazon S3 on Outposts 存储容量的更多信息，请参阅以下主题。

主题

- 为 S3 on Outposts 桶管理 S3 版本控制
- 为 Amazon S3 on Outposts 桶创建和管理生命周期配置
- 复制 S3 on Outposts 的对象
- 通过使用 AWS RAM 共享 S3 on Outposts
- 使用 S3 on Outposts 的其他 AWS 服务

为 S3 on Outposts 桶管理 S3 版本控制

启用后，S3 版本控制功能将对象的多个不同副本保存到同一个桶中。对于 Outposts 桶中存储的每个对象，您可以使用 S3 版本控制功能来保留、检索和还原其每个版本。S3 版本控制功能可帮助您从用户意外操作和应用程序故障中恢复。

Amazon S3 on Outposts 桶具有三种版本控制状态：

- Unversioned（不受版本控制）- 如果您从未在桶上启用或暂停过 S3 版本控制，则桶处于不受版本控制状态，并且不返回 S3 版本控制状态。有关 S3 版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。
- Enabled（已启用）- 为桶中的对象启用 S3 版本控制。添加到桶的所有对象都将收到唯一的版本 ID。启用版本控制时存在于存储桶中的对象的版本 ID 为`null`。如果使用其他操作修改这些（或任何其他）对象，例如 PutObject，则新对象将获得唯一的版本 ID。
- Suspended（暂停）- 对桶中的对象暂停 S3 版本控制。暂停版本控制后添加到桶的所有对象都将收到版本 ID `null`。有关更多信息，请参阅将对象添加到已暂停版本控制的存储桶。

在对 S3 on Outposts 桶启用 S3 版本控制后，它将无法返回到不受版本控制状态。但是，您可以暂停版本控制。有关 S3 版本控制的更多信息，请参阅在 S3 存储桶中使用版本控制。

对于桶中的每个对象，您都有一个当前版本以及零个或零个以上的非当前版本。为了降低存储成本，您可以将桶 S3 生命周期规则配置为使非当前版本在指定时间段后过期。有关更多信息，请参阅为 Amazon S3 on Outposts 桶创建和管理生命周期配置。

以下示例显示如何使用 AWS Management Console 和 AWS Command Line Interface（AWS CLI）对现有 S3 on Outposts 桶启用或暂停版本控制。要创建启用了 S3 版本控制的桶，请参阅创建 S3 on Outposts 存储桶。

> **Note**
>
> 创建存储桶的 AWS 账户 拥有该存储桶，也是唯一可以向其提交操作的账户。存储桶具有配置属性，如 Outpost、标签、默认加密和访问点设置。访问点设置包括用于访问存储桶中对象的 Virtual Private Cloud (VPC) 和访问点策略以及其他元数据。有关更多信息，请参阅S3 on Outposts 规范。

使用 S3 控制台

编辑桶的 S3 版本控制设置

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 存储桶）。
3. 选择要为其启用 S3 版本控制的 Outposts 桶。
4. 选择 Properties（属性）选项卡。
5. 在 Bucket Versioning (存储桶版本控制) 下，请选择 Edit (编辑)。
6. 通过选择以下选项之一，编辑桶的 S3 版本控制设置：
   - 要暂停 S3 版本控制并停止创建新的对象版本，请选择 Suspend（暂停）。
   - 要启用 S3 版本控制并保存每个对象的多个不同副本，请选择 Enable（启用）。
7. 选择保存更改。

使用 AWS CLI

要使用 AWS CLI 对桶启用或暂停 S3 版本控制，请使用 `put-bucket-versioning` 命令，如以下示例所示。要使用这些示例，请将每个 *`user input placeholder`* 替换为您自己的信息。

有关更多信息，请参阅《AWS CLI 参考》中的 put-bucket-versioning。

Example ：启用 S3 版本控制

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket --versioning-configuration Status=Enabled
```

Example ：暂停 S3 版本控制

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket --versioning-configuration Status=Suspended
```

为 Amazon S3 on Outposts 桶创建和管理生命周期配置

您可以使用 S3 生命周期优化 Amazon S3 on Outposts 的存储容量。您可以创建生命周期规则，使对象在老化时过期或被较新版本取代。您可以创建、启用、禁用或删除生命周期规则。

有关 S3 生命周期的更多信息，请参阅管理存储生命周期。

> **Note**
>
> 创建桶的 AWS 账户拥有该桶，并且是唯一可以创建、启用、禁用或删除生命周期规则的账户。

要创建和管理 S3 on Outposts 桶的生命周期配置，请参阅以下主题。

主题

- 使用 AWS Management Console创建和管理生命周期规则
- 使用 AWS CLI 和适用于 Java 的 SDK 创建和管理生命周期配置

使用 AWS Management Console创建和管理生命周期规则

您可以使用 S3 生命周期优化 Amazon S3 on Outposts 的存储容量。您可以创建生命周期规则，使对象在老化时过期或被较新版本取代。您可以创建、启用、禁用或删除生命周期规则。

有关 S3 生命周期的更多信息，请参阅管理存储生命周期。

> **Note**
>
> 创建桶的 AWS 账户拥有该桶，并且是唯一可以创建、启用、禁用或删除生命周期规则的账户。

要使用 AWS Management Console 为 S3 on Outposts 创建和管理生命周期规则，请参阅以下主题。

主题

- 创建生命周期规则
- 启用生命周期规则
- 编辑生命周期规则

- 删除生命周期规则

创建生命周期规则

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 桶）。
3. 选择要为其创建生命周期规则的 Outposts 桶。
4. 选择 Management（管理）选项卡，然后选择 Create Lifecycle rule（创建生命周期规则）。
5. 为 Lifecycle rule name（生命周期规则名称）输入一个值。
6. 在 Rule scope（角色范围）下，选择以下选项之一：
   - 为将范围限制到特定筛选条件，请选择 Limit the scope of this rule using one or more filters（使用一个或多个筛选条件限制此规则的范围）。然后，添加前缀筛选条件、标签或对象大小。
   - 要将此规则应用于桶中的所有对象，请选择 Apply to all objects in the bucket（应用到桶中的所有对象）。
7. 在 Lifecycle rule actions（生命周期规则操作）下，选择以下选项之一：
   - Expire current versions of objects（将对象的当前版本设为过期）- 对于启用版本控制的桶，S3 on Outposts 会添加删除标记，并将对象保留为非当前版本。对于不使用 S3 版本控制的桶，S3 on Outposts 会永久删除对象。
   - Permanently delete noncurrent versions of objects（永久删除对象的非当前版本）– S3 on Outposts 永久删除对象的非当前版本。
   - Delete expired object delete markers or incomplete multipart uploads（删除过期的删除标记或未完成的分段上传）– S3 on Outposts 永久删除过期的删除标记或未完成的分段上传。

     如果您使用对象标签限制生命周期规则的范围，则无法选择 Delete expired object delete markers（删除过期的对象删除标记）。如果您选择 Expire current object versions（使当前对象版本过期），也无法选择 Delete expired object delete markers（删除过期的对象删除标记）。

> **Note**
>
> 基于大小的筛选条件不能用于删除标记和未完成的分段上传。

8. 如果您选择了 Expire current versions of objects（使当前版本的对象过期）或 Permanently delete noncurrent versions of objects（永久删除对象的非当前版本），请根据特定日期或对象的存在期限配置规则触发器。
9. 如果您选择了 Delete expired object delete markers（删除过期的对象删除标记），要确认删除过期的对象删除标记，请选择 Delete expired object delete markers（删除过期的对象删除标记）。
10. 在 Timeline Summary（时间线摘要）下，查看您的生命周期规则，然后选择 Create rule（创建规则）。

启用生命周期规则

启用或禁用桶生命周期规则

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 桶）。
3. 选择要为其启用或禁用生命周期规则的 Outposts 桶。
4. 选择 Management（管理）选项卡，然后在 Lifecycle rule（生命周期规则）下选择要启用或禁用的规则。
5. 对于 Action（操作），选择 Enable or disable rule（启用或禁用规则）。

编辑生命周期规则

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 桶）。
3. 选择要为其编辑生命周期规则的 Outposts 桶。
4. 选择 Management（管理）选项卡，然后选择要编辑的生命周期规则。
5. （可选）更新 Lifecycle rule name（生命周期规则名称）的值。
6. 在 Rule scope（规则范围）下，根据需要编辑范围：
   - 为将范围限制到特定筛选条件，请选择 Limit the scope of this rule using one or more filters（使用一个或多个筛选条件限制此规则的范围）。然后，添加前缀筛选条件、标签或对象大小。
   - 要将此规则应用于桶中的所有对象，请选择 Apply to all objects in the bucket（应用到桶中的所有对象）。
7. 在 Lifecycle rule actions（生命周期规则操作）下，选择以下选项之一：

- Expire current versions of objects（将对象的当前版本设为过期）- 对于启用版本控制的桶，S3 on Outposts 会添加删除标记，并将对象保留为非当前版本。对于不使用 S3 版本控制的桶，S3 on Outposts 会永久删除对象。
- Permanently delete noncurrent versions of objects（永久删除对象的非当前版本）– S3 on Outposts 永久删除对象的非当前版本。
- Delete expired object delete markers or incomplete multipart uploads（删除过期的删除标记或未完成的分段上传）– S3 on Outposts 永久删除过期的删除标记或未完成的分段上传。

  如果您使用对象标签限制生命周期规则的范围，则无法选择 Delete expired object delete markers（删除过期的对象删除标记）。如果您选择 Expire current object versions（使当前对象版本过期），也无法选择 Delete expired object delete markers（删除过期的对象删除标记）。

> **Note**
>
> 基于大小的筛选条件不能用于删除标记和未完成的分段上传。

8. 如果您选择了 Expire current versions of objects（使当前版本的对象过期）或 Permanently delete noncurrent versions of objects（永久删除对象的非当前版本），请根据特定日期或对象存在期限配置规则触发器。
9. 如果您选择了 Delete expired object delete markers（删除过期的对象删除标记），要确认删除过期的对象删除标记，请选择 Delete expired object delete markers（删除过期的对象删除标记）。
10. 选择 Save（保存）。

删除生命周期规则

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 桶）。
3. 选择要为其删除生命周期规则的 Outposts 桶。
4. 选择 Management（管理）选项卡，然后 Lifecycle rule（生命周期规则）下，选择要删除的规则。
5. 选择 Delete。

使用 AWS CLI 和适用于 Java 的 SDK 创建和管理生命周期配置

您可以使用 S3 生命周期优化 Amazon S3 on Outposts 的存储容量。您可以创建生命周期规则，使对象在老化时过期或被较新版本取代。您可以创建、启用、禁用或删除生命周期规则。

有关 S3 生命周期的更多信息，请参阅管理存储生命周期。

> **Note**
>
> 创建桶的 AWS 账户拥有该桶，并且是唯一可以创建、启用、禁用或删除生命周期规则的账户。

要使用 AWS Command Line Interface (AWS CLI) 和 AWS SDK for Java 创建和管理 S3 on Outposts 桶的生命周期配置，请参阅以下示例。

主题

- 放置 (PUT) 生命周期配置
- 获取 (GET) S3 on Outposts 桶上的生命周期配置

放置 (PUT) 生命周期配置

AWS CLI

以下 AWS CLI 示例在 Outposts 桶上放置生命周期配置策略。此策略指定具有标记前缀 (*myprefix*) 的所有对象，并且标签在 10 天后过期。要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。

1. 将生命周期配置策略保存到 JSON 文件中。在本示例中，文件命名为 `lifecycle1.json`。

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
```

```
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ],
                    "ObjectSizeGreaterThan": 1000,
                    "ObjectSizeLessThan": 5000
                }
            },
            "Status": "Enabled",
            "Expiration": {
                "Days": 10
            }
        }
    ]
}
```

2. 将 JSON 文件作为put-bucket-lifecycle-configuration CLI 命令的一部分提交。要使用此命令，请将每个 *user input placeholder* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 put-bucket-lifecycle-configuration。

```
aws s3control put-bucket-lifecycle-configuration --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket --lifecycle-configuration file://lifecycle1.json
```

SDK for Java

以下适用于 Java 的 SDK 示例在 Outposts 桶上放置生命周期配置。此生命周期配置指定具有标记前缀 (*myprefix*) 的所有对象，并且标签在 10 天后过期。要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 PutBucketLifecycleConfiguration。

```
import com.amazonaws.services.s3control.model.*;

public void putBucketLifecycleConfiguration(String bucketArn) {

    S3Tag tag1 = new S3Tag().withKey("mytagkey1").withValue("mytagkey1");
    S3Tag tag2 = new S3Tag().withKey("mytagkey2").withValue("mytagkey2");
```

```
    LifecycleRuleFilter lifecycleRuleFilter = new LifecycleRuleFilter()
            .withAnd(new LifecycleRuleAndOperator()
                    .withPrefix("myprefix")
                    .withTags(tag1, tag2))
                    .withObjectSizeGreaterThan(1000)
                    .withObjectSizeLessThan(5000);

    LifecycleExpiration lifecycleExpiration = new LifecycleExpiration()
            .withExpiredObjectDeleteMarker(false)
            .withDays(10);

    LifecycleRule lifecycleRule = new LifecycleRule()
            .withStatus("Enabled")
            .withFilter(lifecycleRuleFilter)
            .withExpiration(lifecycleExpiration)
            .withID("id-1");

    LifecycleConfiguration lifecycleConfiguration = new LifecycleConfiguration()
            .withRules(lifecycleRule);

    PutBucketLifecycleConfigurationRequest reqPutBucketLifecycle = new
 PutBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withLifecycleConfiguration(lifecycleConfiguration);

    PutBucketLifecycleConfigurationResult respPutBucketLifecycle =
 s3ControlClient.putBucketLifecycleConfiguration(reqPutBucketLifecycle);
    System.out.printf("PutBucketLifecycleConfiguration Response: %s%n",
 respPutBucketLifecycle.toString());

}
```

获取 (GET) S3 on Outposts 桶上的生命周期配置

AWS CLI

以下 AWS CLI 示例获取 Outposts 桶上的生命周期配置。要使用此命令，请将每个 *user input placeholder* 替换为您自己的信息。有关此命令的更多信息，请参阅《AWS CLI 参考》中的 get-bucket-lifecycle-configuration。

```
aws s3control get-bucket-lifecycle-configuration --account-id 123456789012 --bucket
 arn:aws:s3-outposts:<your-region>:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket
```

SDK for Java

以下适用于 Java 的 SDK 示例获取 Outposts 桶的生命周期配置。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 GetBucketLifecycleConfiguration。

```
import com.amazonaws.services.s3control.model.*;

public void getBucketLifecycleConfiguration(String bucketArn) {

    GetBucketLifecycleConfigurationRequest reqGetBucketLifecycle = new
 GetBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    GetBucketLifecycleConfigurationResult respGetBucketLifecycle =
 s3ControlClient.getBucketLifecycleConfiguration(reqGetBucketLifecycle);
    System.out.printf("GetBucketLifecycleConfiguration Response: %s%n",
 respGetBucketLifecycle.toString());

}
```

复制 S3 on Outposts 的对象

借助 AWS Outposts 上的 S3 复制，您可以将 Amazon S3 on Outposts 配置为在不同的 Outposts 之间或在同一 Outpost 上的桶之间自动复制 S3 对象。您可以使用 Outposts 上的 S3 复制，在相同或不同的 Outposts 或跨不同的账户维护数据的多个副本，以帮助满足数据驻留需求。Outposts 上的 S3 复制有助于满足您的合规存储需求并实现跨账户的数据共享。如果需要确保副本与源数据完全相同，可以使用 Outposts 上的 S3 复制来制作保留所有元数据的对象的副本，如原始对象创建时间、标签和版本 ID。

Outposts 上的 S3 复制还提供详细的指标和通知，以监控桶之间对象复制的状态。您可以使用 Amazon CloudWatch，通过跟踪待复制的字节数、待复制的操作数以及源桶和目标桶之间的复制延迟来监控复制进度。要快速诊断和纠正配置问题，还可以设置 Amazon EventBridge 以接收有关复制对象失败的通知。要了解更多信息，请参阅 管理您的复制。

主题

复制配置

S3 on Outposts 以 XML 格式存储复制配置。在复制配置 XML 文件中，您将指定一个 AWS Identity and Access Management (IAM) 角色以及一个或多个规则。

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

S3 on Outposts 无法在未经您许可的情况下复制对象。您使用在复制配置中指定的 IAM 角色向 S3 on Outposts 授予权限。S3 on Outposts 代入此 IAM 角色以代表您复制对象。在开始复制之前，您必须向 IAM 角色授予所需的权限。有关 S3 on Outposts 的这些权限的更多信息，请参阅创建 IAM 角色。

在以下场景的复制配置中添加一个规则：

- 您希望复制所有对象。
- 您希望复制对象子集。通过在规则中添加一个筛选条件，可标识对象子级。在该筛选条件中，指定对象键前缀、标签或二者的组合以标识要向其应用规则的对象子集。

如果要复制其他对象子集，请在复制配置中添加多个规则。在每个规则中，指定一个选择不同对象子集的筛选条件。例如，您可能选择了键前缀为 `tax/` 或 `document/` 的复制对象。要做到这一点，您需要添加两个规则，一个规则指定 `tax/` 键前缀筛选条件，另一个指定 `document/` 键前缀。

有关 S3 on Outposts 复制配置和复制规则的更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 ReplicationConfiguration。

Outposts 上的 S3 复制的要求

复制有下列要求：

- 目标 Outpost CIDR 范围必须在源 Outpost 子网表中关联。有关更多信息，请参阅创建复制规则的先决条件。
- 源桶和目标桶必须均已启用 S3 版本控制。有关版本控制的更多信息，请参阅 为 S3 on Outposts 桶管理 S3 版本控制。
- Amazon S3 on Outposts 必须有权代表您将对象从源桶复制到目标桶。这意味着您必须创建一个服务角色，以便将 GET 和 PUT 权限委派给 S3 on Outposts。
  1. 在创建服务角色之前，您必须拥有源桶的 GET 权限和目标桶的 PUT 权限。
  2. 要创建服务角色以将权限委派给 S3 on Outposts，您必须先配置权限以允许 IAM 实体（用户或角色）执行 `iam:CreateRole` 和 `iam:PassRole` 操作。然后，您允许 IAM 实体创建服务角色。要使 S3 on Outposts 代表您代入此服务角色，并将 GET 和 PUT 权限委派给 S3 on Outposts，您必须为该角色分配必要的信任和权限策略。有关 S3 on Outposts 的这些权限的更多信息，请参阅创建 IAM 角色。有关创建服务角色的更多信息，请参阅创建服务角色。

复制什么内容？

原定设置情况下，S3 on Outposts 会复制以下内容：

- 添加复制配置之后创建的对象。
- 从源对象到副本的对象元数据。有关如何将元数据从副本复制到源对象的信息，请参阅在 Outposts 上启用了 Amazon S3 副本修改同步时的复制状态。
- 对象标签（如果有）。

删除操作对复制操作有何影响

如果您从源桶中删除对象，则默认情况下会执行以下操作：

- 如果您发出 DELETE 请求而未指定对象版本 ID，S3 on Outposts 会添加删除标记。S3 on Outposts 将按如下所示处理该删除标记：
  - 原定设置情况下，S3 on Outposts 不复制删除标记。

  - 但是，您可以将删除标记复制添加到非基于标记的规则。有关如何在复制配置中启用删除标记复制的更多信息，请参阅使用 S3 控制台。
- 如果您在 DELETE 请求中指定一个要删除的对象版本 ID，S3 on Outposts 会在源桶中永久删除该对象版本。但是，它不会将删除操作复制到目标桶中。换句话说，它不会从目标桶中删除同一对象版本。此行为可防止恶意删除数据。

不复制什么内容？

原定设置情况下，S3 on Outposts 不复制以下内容：

- 源桶中作为另一个复制规则所建副本的对象。例如，假设您配置的复制中，桶 A 是源，桶 B 是目标。现在假设您添加另一个复制配置，其中桶 B 是源，而桶 C 是目标。在这种情况下，存储桶 B 中作为存储桶 A 中对象的副本的对象不会复制到存储桶 C。
- 源存储桶中已复制到其他目标的对象。例如，如果您在现有复制配置中更改目标桶，则 S3 on Outposts 不会再次复制对象。
- 使用具有客户提供的加密密钥的服务器端加密（SSE-C）创建的对象。
- 对桶级别子资源进行的更新。

  例如，如果您更改生命周期配置或向源桶添加通知配置，则这些更改不会应用于目标桶。此功能使您可以在源桶和目标桶中具有不同的配置。
- 由生命周期配置执行的操作。

  例如，如果您只在源桶上启用生命周期配置并配置过期操作，S3 on Outposts 会为源桶中的过期对象创建删除标记，但不会将这些标记复制到目标桶。如果您希望对源桶和目标桶应用相同的生命周期配置，请对这两个桶启用相同的生命周期配置。有关生命周期配置的更多信息，请参阅 管理存储生命周期。

Outposts 上的 S3 复制不支持什么？

S3 on Outposts 目前不支持以下 S3 复制功能。

- S3 Replication Time Control（S3 RTC）。不支持 S3 RTC，因为 Outposts 上的 S3 复制中的对象流量通过您的本地网络（本地网关）传输。有关本地网关的更多信息，请参阅《AWS Outposts 用户指南》中的使用本地网关。
- 适用于批量操作的 S3 复制。

设置复制

> **Note**
>
> 将不会自动复制桶中在您设置复制规则之前就存在的对象。换句话说，Amazon S3 on Outposts 不以回溯方式复制对象。要复制在复制配置之前创建的对象，您可以使用 `CopyObject` API 操作将它们复制到同一个桶中。复制对象后，它们在桶中显示为“新”对象，复制配置将应用于它们。有关复制对象的详细信息，请参阅使用 AWS SDK for Java 复制 Amazon S3 on Outposts 存储桶中的对象和《Amazon Simple Storage Service API 参考》中的 CopyObject。

要启用 Outposts 上的 S3 复制，请将复制规则添加到您的源 Outposts 桶。复制规则指示 S3 on Outposts 按照指定的方式复制对象。在复制规则中，您必须提供以下内容：

- 源 Outposts 桶接入点 – 您希望 S3 on Outposts 从中复制对象的桶的接入点 Amazon 资源名称（ARN）或接入点别名。有关使用接入点别名的更多信息，请参阅为 S3 on Outposts 桶接入点使用桶式别名。
- 要复制的对象 – 您可以复制源 Outposts 桶中的所有对象或对象子集。通过在配置中提供一个键名前缀和/或一个或多个对象标签，可标识子集。

  例如，如果您配置复制规则以仅复制键名称前缀为 `Tax/` 的对象，则 S3 on Outposts 仅复制键为 `Tax/doc1` 或 `Tax/doc2` 之类的对象。但它不复制具有键 `Legal/doc3` 的对象。如果您同时指定前缀和一个或多个标签，则 S3 on Outposts 仅复制具有特定键前缀和标签的对象。
- 目标 Outposts 桶 – 您希望 S3 on Outposts 将对象复制到的桶的 ARN 或接入点别名。

您可以使用 REST API、AWS SDK、AWS Command Line Interface（AWS CLI）或 Amazon S3 控制台配置复制规则。

S3 on Outposts 还提供了 API 操作来支持设置复制规则。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的以下主题：

- PutBucketReplication
- GetBucketReplication
- DeleteBucketReplication

主题

- 创建复制规则的先决条件
- 在 Outposts 上创建复制规则

创建复制规则的先决条件

主题

- 连接您的源和目标 Outpost 子网
- 创建 IAM 角色

连接您的源和目标 Outpost 子网

要使复制流量通过本地网关从源 Outpost 传输到目标 Outpost，您必须添加一条新路由来设置网络。必须将接入点的无类别域间路由（CIDR）网络范围连接在一起。对于每对接入点，您只需设置一次此连接。

设置连接的某些步骤有所不同，具体取决于与接入点关联的 Outposts 端点的访问类型。端点的访问类型为私有（适用于 AWS Outposts 的直接虚拟私有云 [VPC] 路由）或客户拥有的 IP（本地网络中客户拥有的 IP 地址池 [CoIP 池]）。

步骤 1：查找源 Outposts 端点的 CIDR 范围

查找与源接入点关联的源端点的 CIDR 范围

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 桶）。
3. 在 Outposts 桶列表中，选择要用于复制的源桶。
4. 选择 Outposts 接入点选项卡，然后为您的复制规则选择源桶的 Outposts 接入点。
5. 选择 Outposts 端点。
6. 复制子网 ID 以在步骤 5 中使用。
7. 用于查找源 Outposts 端点的 CIDR 范围的方法取决于您的端点的访问类型。

   在 Outposts 端点概览部分中，查看访问类型。

   - 如果访问类型为私有，请复制无类别域间路由（CIDR）值以在步骤 6 中使用。
   - 如果访问类型为客户拥有的 IP，请执行以下操作：

1. 复制客户拥有的 IPv4 池值，稍后将用作地址池的 ID。
2. 打开 AWS Outposts 控制台 (https://console.aws.amazon.com/outposts/)。
3. 在导航窗格中，选择本地网关路由表。
4. 选择您的源 Outpost 的本地网关路由表 ID 值。
5. 在详细信息窗格中，选择 CoIP 池选项卡。将您之前复制的 CoIP 池 ID 的值粘贴到搜索框中。
6. 对于匹配的 CoIP 池，复制您的源 Outposts 端点的相应 CIDR 值，以便在步骤 6 中使用。

步骤 2：查找目标 Outposts 端点的子网 ID 和 CIDR 范围

要查找与目标接入点关联的目标端点的子网 ID 和 CIDR 范围，请按照步骤 1 中的相同子步骤进行操作，并在应用这些子步骤时将源 Outposts 端点更改为目标 Outposts 端点。复制目标 Outposts 端点的子网 ID 值，以便在步骤 6 中使用。复制目标 Outposts 端点的 CIDR 值，以便在步骤 5 中使用。

步骤 3：查找源 Outpost 的本地网关 ID

查找源 Outpost 的本地网关 ID

1. 打开 AWS Outposts 控制台 (https://console.aws.amazon.com/outposts/)。
2. 在左侧导航窗格中，选择本地网关。
3. 在本地网关页面上，找到要用于复制的源 Outpost 的 Outpost ID。
4. 复制源 Outpost 的本地网关 ID 值，以便在步骤 5 中使用。

有关本地网关的更多信息，请参阅《AWS Outposts 用户指南》中的本地网关。

步骤 4：查找目标 Outpost 的本地网关 ID

要查找目标 Outpost 的本地网关 ID，请按照步骤 3 中的相同子步骤操作，不同之处是查找目标 Outpost 的 Outpost ID。复制目标 Outpost 的本地网关 ID 值，以便在步骤 6 中使用。

步骤 5：设置从源 Outpost 子网到目标 Outpost 子网的连接

从源 Outpost 子网连接到目标 Outpost 子网

1. 登录到 AWS Management Console并打开 VPC 控制台，网址为：https://console.aws.amazon.com/vpc/。
2. 在左侧导航窗格中，选择 Subnets。

3. 在搜索框中，输入您在步骤 1 中找到的源 Outposts 端点的子网 ID。选择具有匹配的子网 ID 的子网。
4. 对于匹配的子网项目，选择该子网的路由表值。
5. 在包含选定路由表的页面上，选择操作，然后选择编辑路由。
6. 在编辑路由页面上，选择添加路由。
7. 在目标下，输入您在步骤 2 中找到的目标 Outposts 端点的 CIDR 范围。
8. 在目标下，选择 Outpost 本地网关，然后输入您在步骤 3 中找到的源 Outpost 的本地网关 ID。
9. 选择 Save changes（保存更改）。
10. 确保路由的状态为活动。

步骤 6：设置从目标 Outpost 子网到源 Outpost 子网的连接

1. 登录到 AWS Management Console并打开 VPC 控制台，网址为：https://console.aws.amazon.com/vpc/。
2. 在左侧导航窗格中，选择 Subnets。
3. 在搜索框中，输入您在步骤 2 中找到的目标 Outposts 端点的子网 ID。选择具有匹配的子网 ID 的子网。
4. 对于匹配的子网项目，选择该子网的路由表值。
5. 在包含选定路由表的页面上，选择操作，然后选择编辑路由。
6. 在编辑路由页面上，选择添加路由。
7. 在目标下，输入您在步骤 1 中找到的源 Outposts 端点的 CIDR 范围。
8. 在目标下，选择 Outpost 本地网关，然后输入您在步骤 4 中找到的目标 Outpost 的本地网关 ID。
9. 选择 Save changes（保存更改）。
10. 确保路由的状态为活动。

连接源接入点和目标接入点的 CIDR 网络范围后，必须创建 AWS Identity and Access Management（IAM）角色。

创建 IAM 角色

原定设置情况下，所有 S3 on Outposts 资源（桶、对象和相关子资源）都是私有的，只有资源拥有者可以访问相应的资源。S3 on Outposts 需要权限才能从源 Outposts 桶读取和复制对象。您通过创建一个 IAM 服务角色，然后在复制配置中指定该角色，授予这些权限。

此部分介绍信任策略及所需的最低权限策略。示例演练提供创建 IAM 角色的分步说明。有关更多信息，请参阅在 Outposts 上创建复制规则。有关 IAM 角色的更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

- 以下示例显示了一个信任策略，您在其中将 S3 on Outposts 标识为可代入此角色的服务主体。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3-outposts.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

- 以下示例显示了一个访问策略，您在其中向角色授予代表您执行复制任务的权限。当 S3 on Outposts 代入此角色时，它将具有您在此策略中指定的权限。要使用这一策略，请将 *user input placeholders* 替换为您自己的信息。确保将它们替换为源和目标 Outpost 的 Outpost ID 以及源和目标 Outposts 桶的桶名称和接入点名称。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:GetObjectVersionForReplication",
            "s3-outposts:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
bucket/SOURCE-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
accesspoint/SOURCE-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      },
      {
         "Effect":"Allow",
```

```
         "Action":[
            "s3-outposts:ReplicateObject",
            "s3-outposts:ReplicateDelete"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/
bucket/DESTINATION-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/
accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      }
   ]
}
```

访问策略授予以下操作的权限：

- `s3-outposts:GetObjectVersionForReplication` – 授予对所有对象执行此操作的权限，以允许 S3 on Outposts 获取与每个对象关联的特定对象版本。

- `s3-outposts:GetObjectVersionTagging` – 针对 *SOURCE-OUTPOSTS-BUCKET* 桶（源桶）中的对象执行此操作的权限允许 S3 on Outposts 读取对象标签以进行复制。有关更多信息，请参阅为 S3 on Outposts 存储桶添加标签。如果 S3 on Outposts 没有此权限，它将复制对象而不是对象标签。
- `s3-outposts:ReplicateObject` 和 `s3-outposts:ReplicateDelete` – 针对 *DESTINATION-OUTPOSTS-BUCKET* 桶（目标桶）中的所有对象执行这些操作的权限允许 S3 on Outposts 将对象或删除标记复制到目标 Outposts 桶。有关删除标记的信息，请参阅 删除操作对复制操作有何影响。

  > **Note**
  >
  > - 对 *DESTINATION-OUTPOSTS-BUCKET* 桶（目标桶）执行 `s3-outposts:ReplicateObject` 操作的权限还允许复制对象标签。因此，您无需显式授予执行 `s3-outposts:ReplicateTags` 操作的权限。
  > - 对于跨账户复制，目标 Outposts 桶的拥有者必须更新其桶策略，以授予对 *DESTINATION-OUTPOSTS-BUCKET* 执行 `s3-outposts:ReplicateObject` 操作的权限。`s3-outposts:ReplicateObject` 操作允许 S3 on Outposts 将对象和对象标签复制到目标 Otposts 桶。

有关 S3 on Outposts 操作的列表，请参阅 S3 on Outposts 定义的操作。

> **Important**
>
> 拥有 IAM 角色的 AWS 账户 必须拥有其向 IAM 角色授予的操作的权限。
> 例如，假定源 Outposts 桶包含由另一个 AWS 账户拥有的对象。对象的拥有者必须通过桶策略和接入点策略，向拥有 IAM 角色的 AWS 账户显式授予必要的权限。否则，S3 on Outposts 无法访问这些对象，因而这些对象的复制将失败。
> 此处介绍的权限与最低复制配置相关。如果您选择添加可选复制配置，则必须向 S3 on Outposts 授予其他权限。

在源和目标 Outposts 桶由不同的 AWS 账户拥有时授予权限

当源和目标 Outposts 桶由不同的账户拥有时，目标 Outposts 桶的拥有者必须针对目标桶更新桶策略和接入点策略。这些策略必须向源 Outposts 桶的拥有者和 IAM 服务角色授予执行复制操作的权限，如以下策略示例所示，否则复制将失败。在以下策略示例中，*DESTINATION-OUTPOSTS-BUCKET* 是目标桶。要使用这些策略示例，请将 *user input placeholders* 替换为您自己的信息。

如果您手动创建 IAM 服务角色，请将角色路径设置为 `role/service-role/`，如以下策略示例所示。有关更多信息，请参阅《IAM 用户指南》中的 IAM ARN。

```
{
   "Version":"2012-10-17",
   "Id":"PolicyForDestinationBucket",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3-outposts:ReplicateDelete",
            "s3-outposts:ReplicateObject"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:DestinationBucket-account-
ID:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET/object/*"
         ]
      }
```

```
   ]
}
```

```
{
"Version":"2012-10-17",
   "Id":"PolicyForDestinationAccessPoint",
   "Statement":[
      {
         "Sid":"Permissions on objects",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::SourceBucket-account-ID:role/service-role/source-
account-IAM-role"
         },
         "Action":[
            "s3-outposts:ReplicateDelete",
            "s3-outposts:ReplicateObject"
         ],
         "Resource" :[
            "arn:aws:s3-outposts:region:DestinationBucket-account-
ID:outpost/DESTINATION-OUTPOST-ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/
object/*"
         ]
      }
   ]
}
```

> **Note**
>
> 如果源 Outposts 桶中的对象带有标签，请注意以下事项：
> 如果源 Outposts 桶拥有者向 S3 on Outposts 授予执行 `s3-outposts:GetObjectVersionTagging` 和 `s3-outposts:ReplicateTags` 操作的权限来复制对象标签（通过 IAM 角色），则 Amazon S3 将复制标签以及对象。有关 IAM 角色的信息，请参阅 创建 IAM 角色。

在 Outposts 上创建复制规则

Outposts 上的 S3 复制是跨相同或不同 AWS Outposts 中的桶自动、异步复制对象。复制操作会将源 Outposts 桶中新创建的对象和对象更新复制到目标 Outposts 桶。有关更多信息，请参阅复制 S3 on Outposts 的对象。

> **Note**
>
> 将不会复制在设置复制规则之前存在于源 Outposts 桶中的对象。换句话说，S3 on Outposts 不以回溯方式复制对象。要复制在复制配置之前创建的对象，您可以使用 `CopyObject` API 操作将它们复制到同一个桶中。复制对象后，它们在桶中显示为“新”对象，复制配置将应用于它们。有关复制对象的详细信息，请参阅使用 AWS SDK for Java 复制 Amazon S3 on Outposts 存储桶中的对象和《Amazon Simple Storage Service API 参考》中的 CopyObject。

配置复制时，需要向源 Outposts 桶添加复制规则。复制规则定义要复制的源 Outposts 桶对象和将存储复制的对象的目标 Outposts 桶。您可以创建一条规则，以复制桶中的所有对象或具有特定键名前缀和/或一个或多个对象标签的对象子集。目标 Outposts 桶与源 Outposts 桶可以位于同一 Outpost 中，也可以位于不同的 Outpost 中。

对于 S3 on Outposts 复制规则，您必须提供源 Outposts 桶的接入点 Amazon 资源名称（ARN）和目标 Outposts 桶的接入点 ARN，而不是源和目标 Outposts 桶名称。

如果您指定要删除的对象版本 ID，S3 on Outposts 会在源 Outposts 桶中删除该对象版本。但它不会将删除操作复制到目标 Outposts 桶中。换句话说，它不会从目标 Outposts 桶中删除同一对象版本。此行为可防止恶意删除数据。

当您将复制规则添加到 Outposts 桶后，原定设置情况下将启用复制规则，使该规则在您保存它后立即开始发挥作用。

在此示例中，您为位于不同的 Outposts 上且由同一 AWS 账户拥有的源和目标 Outposts 桶设置复制。提供了使用 Amazon S3 控制台、AWS Command Line Interface (AWS CLI) 以及 AWS SDK for Java 和 AWS SDK for .NET 的示例。有关跨账户 Outposts 上的 S3 复制权限的信息，请参阅在源和目标 Outposts 桶由不同的 AWS 账户拥有时授予权限。

有关设置 S3 on Outposts 复制规则的先决条件，请参阅创建复制规则的先决条件。

使用 S3 控制台

当目标 Amazon S3 on Outposts 桶位于与源 Outposts 桶不同的 Outpost 中时，请按照以下步骤配置复制规则。

如果目标 Outposts 桶与源 Outposts 桶位于不同的账户中，您必须向目标 Outposts 桶添加桶策略，以便向源 Outposts 桶账户的拥有者授予向目标 Outposts 桶复制对象的权限。有关更多信息，请参阅在源存储桶和目标存储桶由不同的 AWS 账户 拥有时授予权限。

创建复制规则

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Outposts 桶列表中，选择要用作源桶的桶名称。
3. 选择管理选项卡，向下滚动到复制规则部分，然后选择创建复制规则。
4. 在复制规则名称下，输入规则的名称以帮助稍后识别规则。该名称是必填项，并且它在桶内必须是唯一的。
5. 在状态下，已启用在原定设置情况下处于选中状态。已启用的规则将在您保存它后立即开始工作。如果您想以后再启用该规则，请选择已禁用。
6. 在优先级下，规则的优先级值决定了在存在重叠的规则时要应用哪条规则。当对象包含在多个复制规则的范围内时，S3 on Outposts 使用这些优先级值来避免冲突。原定设置情况下，新规则以最高优先级添加到复制配置中。数字越大，优先级越高。

   要更改规则的优先级，请在保存规则后，从复制规则列表中选择规则名称，选择操作，然后选择编辑优先级。
7. 在源桶下，您可以通过以下选项设置复制源：

   - 要复制整个桶，请选择应用到桶中的所有 对象。
   - 要将前缀或标签筛选应用到复制源，请选择使用一个或多个筛选条件限制此规则的范围。您可以组合前缀和标签。
     - 要复制所有具有相同前缀的对象，请在前缀下方的框中输入前缀。使用前缀筛选条件将复制限制为名称以相同字符串（例如，`pictures`）开头的所有对象。

       如果您输入的前缀是文件夹名称，则必须使用 /（正斜杠）作为最后一个字符（例如，`pictures/`）。
     - 要复制具有一个或多个相同对象标签的所有对象，请选择添加标签，然后在框中输入键值对。要添加另一个标签，请重复该过程。有关对象标签的更多信息，请参阅 为 S3 on Outposts 存储桶添加标签。
8. 要访问您的 S3 on Outposts 源桶以进行复制，请在源接入点名称下，选择一个连接到源桶的接入点。

9. 在目标下，选择您希望 S3 on Outposts 向其中复制对象的目标 Outposts 桶的接入点 ARN。目标 Outposts 桶可以与源 Outposts 桶位于相同或不同的 AWS 账户中。

   如果目标桶与源 Outposts 桶位于不同的账户中，您必须向目标 Outposts 桶添加桶策略，以便为源 Outposts 桶账户的拥有者授予将对象复制到目标 Outposts 桶中的权限。有关更多信息，请参阅在源和目标 Outposts 桶由不同的 AWS 账户拥有时授予权限。

   > **Note**
   >
   > 如果未对目标 Outposts 桶启用版本控制，您将收到包含启用版本控制按钮的警告。选择此按钮可对桶启用版本控制。

10. 设置 AWS Identity and Access Management（IAM）服务角色，S3 on Outposts 可以代入该角色以代表您复制对象。

    要设置 IAM 角色，请在 IAM 角色下执行以下操作之一：

    - 要让 S3 on Outposts 为复制配置创建新的 IAM 角色，请选择从现有 IAM 角色中选择，然后选择创建新角色。当您保存该规则后，将为 IAM 角色生成一个与您选择的源和目标 Outposts 桶匹配的新策略。建议您选择创建新角色。
    - 还可以选择使用现有的 IAM 角色。在这种情况下，您必须选择一个可以授予 S3 on Outposts 必要权限以进行复制的角色。如果此角色没有授予 S3 on Outposts 足够的权限来遵循您的复制规则，则复制将失败。

      要选择现有角色，请选择从现有 IAM 角色中选择，然后从下拉菜单中选择角色。您也可以选择输入 IAM 角色 ARN，然后输入 IAM 角色的 Amazon 资源名称（ARN）。

    > **Important**
    >
    > 当您将复制规则添加到 S3 on Outposts 桶时，您必须具有 `iam:CreateRole` 和 `iam:PassRole` 权限才能创建和传递向 S3 on Outposts 授予复制权限的 IAM 角色。有关更多信息，请参阅《IAM 用户指南》中的向用户授予将角色传递给 AWS 服务的权限。

11. 原定设置情况下，Outposts 桶中的所有对象均加密。有关 S3 on Outposts 加密的更多信息，请参阅S3 on Outposts 中的数据加密。只能复制使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）进行加密的对象。不支持复制使用具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）或使用客户提供的加密密钥的服务器端加密（SSE-C）进行加密的对象。

12. 根据需要，在设置复制规则配置时启用以下附加选项：

    - 如果要在复制配置中启用 S3 on Outposts 复制指标，请选择复制指标。有关更多信息，请参阅使用复制指标监控进度。
    - 如果要在复制配置中启用删除标记复制，请选择 Delete marker replication (删除标记复制)。有关更多信息，请参阅删除操作对复制操作有何影响。
    - 如果要将对副本所做的元数据更改复制回源对象，请选择副本修改同步。有关更多信息，请参阅在 Outposts 上启用了 Amazon S3 副本修改同步时的复制状态。

13. 要完成操作，请选择创建规则。

当您保存规则之后，您可以编辑、启用、禁用或删除您的规则。要执行这些操作，请转到源 Outposts 桶的管理选项卡，向下滚动到复制规则部分，选择您的规则，然后选择编辑规则。

使用 AWS CLI

要使用 AWS CLI 在源和目标 Outposts 桶由同一 AWS 账户拥有时设置复制，请执行以下操作：

- 创建源和目标 Outposts 桶。
- 对这两个桶启用版本控制。
- 创建向 S3 on Outposts 授予复制对象的权限的 IAM 角色。
- 将复制配置添加到源 Outposts 桶中。

要验证您的设置，请对其进行测试。

当源和目标 Outposts 桶由同一个 AWS 账户拥有时设置复制

1. 为 AWS CLI 设置凭证配置文件。在此示例中，我们使用配置文件名称 `acctA`。有关设置凭证配置文件的信息，请参阅《AWS Command Line Interface 用户指南》中的命名配置文件。

   > **Important**
   >
   > 用于执行此操作的配置文件必须具有必要的权限。例如，在复制配置中，可以指定 S3 on Outposts 可代入的 IAM 服务角色。仅当您所使用的配置文件具有 `iam:CreateRole` 和 `iam:PassRole` 权限时，您才能执行此操作。有关更多信息，请参阅《IAM 用户指南》中的向用户授予将角色传递给 AWS 服务的权限。如果您使用管理员凭证创建命名配置文件，命名配置文件将具有执行所有任务的必要权限。

2. 创建一个源存储桶，并对其启用版本控制。以下 create-bucket 命令在美国东部（弗吉尼亚州北部）（us-east-1）区域中创建一个 *SOURCE-OUTPOSTS-BUCKET* 桶。要使用此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control create-bucket --bucket SOURCE-OUTPOSTS-BUCKET --outpost-id SOURCE-OUTPOST-ID --profile acctA --region us-east-1
```

以下 put-bucket-versioning 命令对于 *SOURCE-OUTPOSTS-BUCKET* 桶启用版本控制。要使用此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
```

3. 创建一个目标存储桶，并对其启用版本控制。以下 create-bucket 命令在美国西部（俄勒冈州）（us-west-2）区域中创建一个 *DESTINATION-OUTPOSTS-BUCKET* 桶。要使用此命令，请将 *user input placeholders* 替换为您自己的信息。

> **Note**
>
> 要在源和目标 Outposts 桶位于同一 AWS 账户中时设置复制配置，请使用同一命名配置文件。此示例使用 acctA。要在两个桶由不同 AWS 账户拥有时对复制配置进行测试，您需要为每个桶指定不同的配置文件。

```
aws s3control create-bucket --bucket DESTINATION-OUTPOSTS-BUCKET --create-bucket-configuration LocationConstraint=us-west-2 --outpost-id DESTINATION-OUTPOST-ID --profile acctA --region us-west-2
```

以下 put-bucket-versioning 命令对于 *DESTINATION-OUTPOSTS-BUCKET* 桶启用版本控制。要使用此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
```

4. 创建 IAM 服务角色。稍后在复制配置中，将此服务角色添加到 *SOURCE-OUTPOSTS-BUCKET* 桶中。S3 on Outposts 代入此角色以代表您复制对象。分两个步骤创建 IAM 角色：

a. 创建一个 IAM 角色。

i. 复制以下信任策略，并将其保存到本地计算机上当前目录中一个名为 s3-on-outposts-role-trust-policy.json 的文件。此策略向 S3 on Outposts 服务主体授予代入该服务角色的权限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3-outposts.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

ii. 运行以下 命令以创建角色。将 *user input placeholders* 替换为您自己的信息。

```
aws iam create-role --role-name replicationRole --assume-role-policy-
document file://s3-on-outposts-role-trust-policy.json --profile acctA
```

b. 将权限策略附加到服务角色。

i. 复制以下权限策略，并将其保存到本地计算机上当前目录中一个名为 s3-on-outposts-role-permissions-policy.json 的文件。此策略授予对各种 S3 on Outposts 桶和对象操作的权限。要使用这一策略，请将 *user input placeholders* 替换为您自己的信息。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:GetObjectVersionForReplication",
            "s3-outposts:GetObjectVersionTagging"
         ],
         "Resource":[
```

```
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-
OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-
OUTPOST-ID/accesspoint/SOURCE-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3-outposts:ReplicateObject",
            "s3-outposts:ReplicateDelete"
         ],
         "Resource":[
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-
OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET/object/*",
            "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-
OUTPOST-ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
         ]
      }
   ]
}
```

ii. 运行以下命令以创建策略并将其附加到角色。将 *user input placeholders* 替换为您自己的信息。

```
aws iam put-role-policy --role-name replicationRole --policy-
document file://s3-on-outposts-role-permissions-policy.json --policy-
name replicationRolePolicy --profile acctA
```

5. 将复制配置添加到 *SOURCE-OUTPOSTS-BUCKET* 桶中。

a. 尽管 S3 on Outposts API 要求复制配置采用 XML 格式，但 AWS CLI 要求您以 JSON 格式指定复制配置。将以下 JSON 保存到计算机上本地目录中一个名为 `replication.json` 文件。要使用此命令，请将 *user input placeholders* 替换为您自己的信息。

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
```

```
      "Destination": {
        "Bucket":
        "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-
ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT"
      }
    }
  ]
}
```

b. 运行以下 put-bucket-replication 命令以将复制配置添加到源 Outposts 桶中。要使用此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control put-bucket-replication --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-
ID/bucket/SOURCE-OUTPOSTS-BUCKET --replication-configuration file://
replication.json --profile acctA
```

c. 要检索复制配置，请使用 get-bucket-replication 命令。要使用此命令，请将 *user input placeholders* 替换为您自己的信息。

```
aws s3control get-bucket-replication --account-id 123456789012 --bucket
 arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
bucket/SOURCE-OUTPOSTS-BUCKET --profile acctA
```

6. 在 Amazon S3 控制台中测试该设置：

a. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

b. 在 *SOURCE-OUTPOSTS-BUCKET* 桶中，创建一个名为 Tax 的文件夹。

c. 将示例对象添加到 *SOURCE-OUTPOSTS-BUCKET* 桶中的 Tax 文件夹。

d. 在 *DESTINATION-OUTPOSTS-BUCKET* 桶中，确认以下几点：

- S3 on Outposts 复制了对象。

> **Note**
>
> S3 on Outposts 复制对象所需的时间量取决于对象大小。有关如何查看复制状态的信息，请参阅获取复制状态信息。

- 在对象的属性选项卡中，复制状态设置为副本（将其标识为副本对象）。

管理您的复制

本节介绍 S3 on Outposts 上可用的其他复制配置选项、如何确定复制状态以及如何排查复制问题。有关核心复制配置的信息，请参阅设置复制。

主题

使用复制指标监控进度

Outposts 上的 S3 复制为复制配置中的复制规则提供了详细的指标。使用复制指标，您可以通过跟踪待复制的字节数、复制延迟的复制和待处理的操作数，以 5 分钟为间隔监控复制的进度。为了帮助排查任何配置问题，您还可以设置 Amazon EventBridge 来接收有关复制失败的通知。

启用复制指标后，Outposts 上的 S3 复制会将以下指标发布到 Amazon CloudWatch：

- 待复制的字节数 – 给定复制规则的待复制对象的总字节数。
- 复制延迟 – 对于给定的复制规则，复制目标桶落后于源桶的最大秒数。
- 待复制的操作 – 给定复制规则的等待复制的操作的数量。操作包括对象、删除标记和标签。

> **Note**
>
> Outposts 上的 S3 复制指标的费率与 CloudWatch 自定义指标费率相同。有关更多信息，请参阅 CloudWatch 定价。

获取复制状态信息

复制状态可以帮助您确定 Amazon S3 on Outposts 正在复制的对象的当前状态。源对象的复制状态将返回 `PENDING`、`COMPLETED`、或 `FAILED`。副本的复制状态将返回 `REPLICA`。

复制状态概述

在复制场景中，您有一个源桶（对它配置复制）和一个目标桶（S3 on Outposts 将对象复制到其中）。当您请求这些桶中的对象（使用 `GetObject`）或对象元数据（使用 `HeadObject`）时，S3 on Outposts 将在响应中返回 `x-amz-replication-status` 标头，如下所示：

- 如果请求源桶中的对象，S3 on Outposts 将返回 `x-amz-replication-status` 标头（如果请求中的对象符合复制条件）。

  例如，假设您在复制配置中指定了对象前缀 `TaxDocs`，以指示 S3 on Outposts 仅复制键名称前缀为 `TaxDocs` 的对象。您上传的具有此键名前缀的任何对象（例如 `TaxDocs/document1.pdf`）都将复制。对于具有此键名称前缀的对象请求，S3 on Outposts 会返回其对象复制状态为以下值之一的 `x-amz-replication-status` 标头：`PENDING`、`COMPLETED` 或 `FAILED`。

  > **Note**
  >
  > 如果在上传对象后对象复制失败，您无法重试复制。您必须重新上传对象。由于缺少复制角色权限或缺少桶权限等问题，对象将转换为 `FAILED` 状态。对于临时故障（例如，如果桶或 Outpost 不可用），复制状态不会转换为 `FAILED`，而是保持 `PENDING`。在资源恢复联机后，S3 on Outposts 将恢复复制这些对象。

- 当请求目标桶中的对象时，如果请求中的对象是 S3 on Outposts 创建的副本，S3 on Outposts 会返回值为 `REPLICA` 的 `x-amz-replication-status` 标头。

> **Note**
>
> 在从启用了复制的源桶中删除对象之前，请检查该对象的复制状态，以确保已复制了该对象。

在 Outposts 上启用了 Amazon S3 副本修改同步时的复制状态

当您的复制规则启用了 S3 on Outposts 副本修改同步时，副本可能报告 `REPLICA` 以外的状态。如果元数据更改正在复制过程中，副本的 `x-amz-replication-status` 标头将返回 `PENDING`。如果副本修改同步无法复制元数据，副本的标头将返回 `FAILED`。如果正确复制元数据，则副本的标头将返回值 `REPLICA`。

对复制进行问题排查

如果在您配置复制之后，对象副本未出现在目标 Amazon S3 on Outposts 桶中，请使用以下问题排查提示确定并修复问题。

- S3 on Outposts 复制对象所需的时间取决于多个因素，包括源和目标 Outposts 之间的距离，以及对象的大小。

  您可以检查源对象的复制状态。如果对象的复制状态为 `PENDING`，表示 S3 on Outpost 尚未完成复制。如果对象的复制状态为 `FAILED`，则应检查对源桶设置的复制配置。
- 在源桶上的复制配置中，验证以下几点：
  - 目标桶的接入点 Amazon 资源名称（ARN）是正确的。
  - 键名前缀是否正确。例如，如果将配置设置为复制具有前缀 `Tax` 的对象，则仅复制具有 `Tax/document1` 或 `Tax/document2` 等键名的对象。不会复制具有键名 `document3` 的对象。
  - 状态是否为 `Enabled`。
- 验证没有在任何一个桶上暂停版本控制。源桶和目标桶必须均已启用版本控制。
- 如果目标桶由另一个 AWS 账户拥有，请验证桶拥有者是否对目标桶设置了允许源桶拥有者复制对象的桶策略。有关示例，请参阅 在源和目标 Outposts 桶由不同的 AWS 账户拥有时授予权限。
- 如果对象副本未出现在目标桶中，以下问题可能会阻止复制：
  - 如果源桶中的某对象是另一个复制配置所创建的副本，S3 on Outposts 不会复制该对象。例如，如果您设置从桶 A 到桶 B 再到桶 C 的复制配置，则 S3 on Outposts 不会将桶 B 中的对象副本复制到桶 C。

    如果您想将桶 A 中的对象复制到桶 B 和桶 C，请在不同的复制规则中为源桶复制配置设置多个桶目标。例如，在源桶 A 上创建两条复制规则，其中一条规则旨在复制到目标桶 B，另一条规则旨在复制到目标桶 C。
  - 源桶拥有者可以向其他 AWS 账户授予上载对象的权限。默认情况下，源桶拥有者对于其他账户创建的对象没有权限。复制配置仅复制源桶拥有者对其具有访问权限的对象。为避免复制问题，源桶拥有者可以向其他 AWS 账户授予有条件地创建对象的权限，从而要求针对这些对象的显式访问权限。有关策略示例，请参阅 在授予上传对象的跨账户权限的同时，确保存储桶拥有者拥有完全控制权。
- 假设在复制配置中，您添加了一个规则以复制具有特定标签的对象子集。在这种情况下，您必须在创建对象时分配特定的标签键和值，以便 S3 on Outposts 复制对象。如果您先创建对象，然后向该现有对象添加标签，S3 on Outposts 将不会复制对象。
- 如果桶策略拒绝针对以下任一操作访问复制角色，则复制将失败：

源桶：

```
"s3-outposts:GetObjectVersionForReplication",
"s3-outposts:GetObjectVersionTagging"
```

目标桶：

```
"s3-outposts:ReplicateObject",
"s3-outposts:ReplicateDelete",
"s3-outposts:ReplicateTags"
```

- 当对象无法复制到其目标 Outposts 时，Amazon EventBridge 可以通知您。有关更多信息，请参阅将 EventBridge 用于 Outposts 上的 S3 复制。

将 EventBridge 用于 Outposts 上的 S3 复制

Amazon S3 on Outposts 已与 Amazon EventBridge 集成，并使用 `s3-outposts` 命名空间。EventBridge 是一种无服务器事件总线服务，您可以使用它将应用程序与来自各种来源的数据相连接。有关更多信息，请参阅 Amazon EventBridge 用户指南中的什么是 Amazon EventBridge？。

为了帮助解决任何复制配置问题，您可以设置 Amazon EventBridge 来接收有关复制失败事件的通知。在对象未复制到其目标 Outposts 的情况下，EventBridge 可以通知您。有关正在复制的对象的当前状态的更多信息，请参阅复制状态概述。

每当 Outposts 桶中发生某些事件时，S3 on Outposts 都可以将事件发送到 EventBridge。与其他目标不同，您无需选择要提供的事件类型。还可以使用 EventBridge 规则将事件路由到其他目标。启用 EventBridge 后，S3 on Outposts 会将以下所有事件发送到 EventBridge。

事件类型	描述	命名空间
`Operation FailedReplication`	复制规则内的对象复制已失败。有关 Outposts 上的 S3 复制失败原因的更多信息，请参阅使用 EventBridge 查看 Outposts 上的 S3 复制失败原因。	`s3-outposts`

使用 EventBridge 查看 Outposts 上的 S3 复制失败原因

下表列出了 Outposts 上的 S3 复制失败原因。您可以将 EventBridge 规则配置为通过 Amazon Simple Queue Service（Amazon SQS）、Amazon Simple Notification Service（Amazon SNS）、AWS Lambda 或 Amazon CloudWatch Logs 发布和查看失败原因。有关将这些资源用于 EventBridge 所需权限的更多信息，请参阅将基于资源的策略用于 EventBridge。

复制失败原因	描述
`AssumeRoleNotPermitted`	S3 on Outposts 无法代入在复制配置中指定的 AWS Identity and Access Management（IAM）角色。
`DstBucketNotFound`	S3 on Outposts 找不到在复制配置中指定的目标桶。
`DstBucketUnversioned`	未在 Outposts 目标桶上启用版本控制。要使用 Outposts 上的 S3 复制来复制对象，必须在目标桶上启用版本控制。
`DstDelObjNotPermitted`	S3 on Outposts 无法将删除操作复制到目标桶。针对目标桶的 `s3-outposts:ReplicateDelete` 权限可能缺失。
`DstMultipartCompleteNotPermitted`	S3 on Outposts 无法在目标桶中完成对象的分段上传。针对目标桶的 `s3-outposts:ReplicateObject` 权限可能缺失。
`DstMultipartInitNotPermitted`	S3 on Outposts 无法启动将对象分段上传到目标桶。针对目标桶的 `s3-outposts:ReplicateObject` 权限可能缺失。
`DstMultipartPartUploadNotPermitted`	S3 on Outposts 无法在目标桶中上传多分段对象。针对目标桶的 `s3-outposts:ReplicateObject` 权限可能缺失。
`DstOutOfCapacity`	因为目标 Outpost 的 S3 存储容量不足，所以 S3 on Outposts 无法复制到此 Outpost。

复制失败原因	描述
DstPutObjNotPermitted	S3 on Outposts 无法将对象复制到目标桶。针对目标桶的 s3-outposts:ReplicateObject 权限可能缺失。
DstPutTaggingNotPermitted	S3 on Outposts 无法将对象标签复制到目标桶。针对目标桶的 s3-outposts:ReplicateObject 权限可能缺失。
DstVersionNotFound	S3 on Outposts 无法在目标桶中找到所需的对象版本，从而无法复制该对象版本的元数据。
SrcBucketReplicationConfigMissing	S3 on Outposts 找不到与源 Outposts 桶关联的接入点的复制配置。
SrcGetObjNotPermitted	S3 on Outposts 无法访问源桶中的对象来进行复制。针对源桶的 s3-outposts:GetObjectVersionForReplication 权限可能缺失。
SrcGetTaggingNotPermitted	S3 on Outposts 无法访问源桶中的对象标签信息。针对源桶的 s3-outposts:GetObjectVersionTagging 权限可能缺失。
SrcHeadObjectNotPermitted	S3 on Outposts 无法从源桶检索对象元数据。针对源桶的 s3-outposts:GetObjectVersionForReplication 权限可能缺失。
SrcObjectNotEligible	对象不符合复制条件。对象或其对象标签与复制配置不匹配。

有关排查复制问题的更多信息，请参阅以下主题：

- 创建 IAM 角色
- 对复制进行问题排查

使用 CloudWatch 监控 EventBridge

为进行监控，Amazon EventBridge 与 Amazon CloudWatch 相集成。EventBridge 会自动每分钟向 CloudWatch 发送指标。这些指标包括规则匹配的事件数以及规则调用目标的次数。当规则在 EventBridge 中运行时，将调用与该规则关联的所有目标。您可以按照以下方式通过 CloudWatch 监控 EventBridge 行为。

- 您可以从 CloudWatch 控制面板监控 EventBridge 规则的可用 EventBridge 指标。然后，您可以使用 CloudWatch 功能（例如 CloudWatch 告警）对某些指标设置告警。如果这些指标达到您在告警中指定的自定义阈值，您将收到通知并可以相应地采取措施。
- 可以将 Amazon CloudWatch Logs 设置为 EventBridge 规则的目标。然后，EventBridge 创建日志流，而 CloudWatch Logs 将事件中的文本存储为日志条目。有关更多信息，请参阅 EventBridge 和 CloudWatch Logs。

有关调试 EventBridge 事件交付和存档事件的更多信息，请参阅以下主题：

- 事件重试策略和使用死信队列
- 存档 EventBridge 事件

通过使用 AWS RAM 共享 S3 on Outposts

Amazon S3 on Outposts 支持通过使用 AWS Resource Access Manager (AWS RAM) 跨一个企业内的多个账户共享 S3 容量。借助 S3 on Outposts 共享，您可以允许其他人在您的 Outpost 上创建和管理存储桶、端点和访问点。

本主题演示如何使用 AWS RAM 与您的 AWS 企业中的另一个 AWS 账户 共享 S3 on Outposts 和相关资源。

先决条件

- Outpost 拥有者账户在 AWS Organizations 中配置了一个企业。有关更多信息，请参阅 AWS Organizations 用户指南中的创建企业。
- 该企业包括您想与之共享 S3 on Outposts 容量的 AWS 账户。有关更多信息，请参阅 AWS Organizations 用户指南中的向 AWS 账户 发送邀请。
- 请选择要共享的以下选项之一。必须选择第二个资源（Subnets（子网）或 Outposts），以便端点也可供访问。端点是网络要求，以便访问 S3 on Outposts 中存储的数据。

选项 1	选项 2
S3 on Outposts 允许用户在您的 Outposts 和访问点上创建存储桶，并将对象添加到这些存储桶。 子网 允许用户使用您的 Virtual Private Cloud (VPC) 和与您的子网关联的端点。	S3 on Outposts 允许用户在您的 Outposts 和访问点上创建存储桶，并将对象添加到这些存储桶。 Outposts 允许用户查看 S3 容量图表和 AWS Outposts 控制台主页。还允许用户在共享的 Outposts 上创建子网并创建端点。

过程

1. 使用拥有 Outpost 的 AWS 账户登录到 AWS Management Console，然后打开 AWS RAM 控制台，地址为 https://console.aws.amazon.com/ram。
2. 确保您已在 AWS RAM 中启用与 AWS Organizations 共享。有关更多信息，请参阅 AWS RAM 用户指南中的在 AWS Organizations 中启用资源共享。
3. 使用先决条件中的选项 1 或选项 2 创建资源共享。如果您有多个 S3 on Outposts 资源，请选择要共享的资源的 Amazon 资源名称 (ARN)。要启用端点，请共享您的子网或 Outpost。

   有关如何创建资源共享的更新信息，请参阅 AWS RAM 用户指南中的创建资源共享。
4. 您与之共享资源的 AWS 账户 现在应该能够使用 S3 on Outposts 了。根据您在先决条件中选择的选项，向账户用户提供以下信息：

选项 1	选项 2
Outpost ID VPC ID 子网 ID 安全组 ID	Outpost ID

> **Note**
>
> 用户可以使用 AWS RAM 控制台、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 确认已与他们共享资源。用户可以使用 get-resource-shares CLI 命令查看他们的现有资源共享。

用法示例

在您与另一个账户共享 S3 on Outposts 资源后，该账户可以管理 Outpost 上的存储桶和对象。如果您共享了 Subnets（子网）资源，那么该账户可以使用您创建的端点。以下示例演示了用户如何使用 AWS CLI 在您共享这些资源后与您的 Outpost 互动。

Example ：创建存储桶

以下示例在 Outpost *op-01ac5d28a6a232904* 上创建名为 *amzn-s3-demo-bucket1* 的存储桶。在使用此命令之前，请针对您的使用案例将每个 *user input placeholder* 替换为适合的值。

```
aws s3control create-bucket --bucket amzn-s3-demo-bucket1 --outpost-
id op-01ac5d28a6a232904
```

有关此命令的更多信息，请参阅《AWS CLI 命令参考》中的 create-bucket。

Example ：创建访问点

以下示例使用下表中的示例参数在 Outpost 上创建访问点。在使用此命令之前，请针对您的使用案例将这些 *user input placeholder* 值和 AWS 区域 代码替换为适合的值。

参数	值
帐户 ID	*111122223333*
访问点名称	*example-outpost-access-point*
Outpost ID	*op-01ac5d28a6a232904*
Outpost 存储桶名称	*amzn-s3-demo-bucket1*
VPC ID	*vpc-1a2b3c4d5e6f7g8h9*

> **Note**
>
> 账户 ID 参数必须是存储桶拥有者的 AWS 账户 ID，即共享用户。

```
aws s3control create-access-point --account-id 111122223333 --name example-outpost-access-point \
--bucket arn:aws:s3-outposts:us-east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket1 \
--vpc-configuration VpcId=vpc-1a2b3c4d5e6f7g8h9
```

有关此命令的更多信息，请参阅《AWS CLI 参考》中的 create-access-point 。

Example ：上传对象

以下示例将通过 AWS 账户 *111122223333* 拥有的 Outpost *op-01ac5d28a6a232904* 上的访问点 *example-outpost-access-point*，将文件 *my_image.jpg* 从用户的本地文件系统上传到名为 *images/my_image.jpg* 的对象。在使用此命令之前，请针对您的使用案例将这些 *user input placeholder* 值和 AWS 区域 代码替换为适合的值。

```
aws s3api put-object --bucket arn:aws:s3-outposts:us-east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-point \
--body my_image.jpg --key images/my_image.jpg
```

有关此命令的更多信息，请参阅《AWS CLI 参考》中的 put-object。

> **Note**
>
> 如果此操作导致未找到资源错误或无响应，则您的 VPC 可能没有共享端点。
> 要检查是否存在共享端点，请使用 list-shared-endpoints AWS CLI 命令。如果没有共享端点，请与 Outpost 拥有者合作创建一个。有关更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 ListSharedEndpoints。

Example ：创建端点

以下示例在共享的 Outpost 上创建端点。在使用此命令之前，请针对您的使用案例将 Outpost ID、子网 ID 和安全组 ID 的 *user input placeholder* 值替换为适合的值。

> **Note**
>
> 只有在资源共享中包含 Outposts 资源时，用户才可以执行此操作。

```
aws s3outposts create-endpoint --outposts-id op-01ac5d28a6a232904 --subnet-id XXXXXX --
security-group-id XXXXXXX
```

有关此命令的更多信息，请参阅《AWS CLI 参考》中的 create-endpoint。

使用 S3 on Outposts 的其他 AWS 服务

在您的 AWS Outposts 本地运行的其他 AWS 服务也可以使用 Amazon S3 on Outposts 容量。在 Amazon CloudWatch 中，S3Outposts 命名空间显示 S3 on Outposts 中存储桶的详细指标，但这些指标不包括其他 AWS 服务的使用情况。要管理其他 AWS 服务使用的 S3 on Outposts 容量，请参阅下表中的信息。

AWS 服务	描述	了解更多信息
Amazon S3	所有直接使用 S3 on Outposts 的情况都具有匹配的账户和存储桶 CloudWatch 指标。	请参阅指标
Amazon Elastic Block Store (Amazon EBS)	对于 Amazon EBS on Outposts，您可以选择 AWS Outpost 作为快照目的地，并将其存储在 S3 on Outpost 本地。	了解更多。
Amazon Relational Database Service (Amazon RDS)	您可以使用 Amazon RDS 本地备份将 RDS 备份存储在 Outpost 本地。	了解更多。

监控 S3 on Outposts

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (OUTPOSTS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的

多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud（VPC）使用接入点和端点连接与 Outposts 桶进行通信。您可以像在 Amazon S3 桶中一样在 Outpost 桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅 什么是 Amazon S3 on Outposts？

有关监控 Amazon S3 on Outposts 存储容量的更多信息，请参阅以下主题。

主题

- 使用 Amazon CloudWatch 指标管理 S3 on Outposts 容量
- 使用 Amazon CloudWatch Events 接收 S3 on Outposts 事件通知
- 使用 AWS CloudTrail 日志监控 S3 on Outposts

使用 Amazon CloudWatch 指标管理 S3 on Outposts 容量

为帮助管理 Outpost 上的固定 S3 容量，我们建议您创建 CloudWatch 提示，在存储利用率超过特定阈值时向您发出提示。有关 S3 on Outposts 的 CloudWatch 指标的更多信息，请参阅CloudWatch 指标。如果没有足够的空间在您的 Outpost 上存储对象，API 将返回容量不足异常 (ICE)。要释放空间，您可以创建触发显式数据删除的 CloudWatch 告警，或者使用生命周期过期策略使对象过期。为在删除数据之前保存数据，您可以使用 AWS DataSync 将数据从 Amazon S3 on Outposts 桶复制到 AWS 区域中的 S3 桶。有关使用 DataSync 的更多信息，请参阅《AWS DataSync 用户指南》中的 AWS DataSync 入门。

CloudWatch 指标

`S3Outposts` 命名空间包括 Amazon S3 on Outposts 桶的以下指标。您可以监控给定桶的总预置 S3 on Outposts 字节数、适用于对象的总可用字节数以及所有对象的总大小。所有直接使用 S3 的情况都存在与桶或账户相关的指标。间接使用 S3（例如在 Outpost 上存储 Amazon Elastic Block Store 本地快照或Amazon Relational Database Service 备份）会消耗 S3 容量，但不包括在桶或账户相关指标中。有关 Amazon EBS 本地快照的更多信息，请参阅 Outposts 上的 Amazon EBS 本地快照。要查看您的 Amazon EBS 成本报告，请访问 https://console.aws.amazon.com/billing/。

> **Note**
>
> S3 on Outposts 仅支持以下指标，而不支持其他 Amazon S3 指标。
> 由于 S3 on Outposts 具有固定容量限制，因此我们建议您创建 CloudWatch 告警，以便在存储利用率超过特定阈值时向您发出通知。

指标	描述	时段	单位	类型
Outpost talByte	Outpost 的预置总容量（以字节为单位）。	5 分钟	字节	S3 on Outposts
Outpost eeBytes	Outpost 上可用于存储客户数据的可用字节数。	5 分钟	字节	S3 on Outposts
BucketU dBytes	给定桶的所有对象的总大小。	5 分钟	字节	S3 on Outposts。仅限直接使用 S3。
Accountl edBytes	指定的 Outposts 账户中所有对象的总大小。	5 分钟	字节	S3 on Outposts。仅限直接使用 S3。
BytesPe ingRepl ation	给定复制规则的待复制对象的总字节数。有关如何启用复制指标的更多信息，请参阅在 Outposts 之间创建复制规则。	5 分钟	字节	可选。适用于 Outposts 上的 S3 复制。
Operati sPendin eplicat n	给定复制规则的待复制操作的总数。有关如何启用复制指标的更多信息，请参阅在 Outposts 之间创建复制规则。	5 分钟	计数	可选。适用于 Outposts 上的 S3 复制。
Replica onLaten	对于给定的复制规则，复制目标桶落后于源桶的当前延迟秒数。有关如何启用复制指标的更多信息，请参阅在 Outposts 之间创建复制规则。	5 分钟	秒	可选。适用于 Outposts 上的 S3 复制。

使用 Amazon CloudWatch Events 接收 S3 on Outposts 事件通知

您可以使用 CloudWatch Events 为任何 Amazon S3 on Outposts API 事件创建规则。创建规则时，您可以选择通过所有支持的 CloudWatch 目标获得通知，包括 Amazon Simple Queue Service（Amazon SQS）、Amazon Simple Notification Service（Amazon SNS）和 AWS Lambda。有关更多信息，请参阅《Amazon CloudWatch Events 用户指南》中的可以作为 CloudWatch Events 目标的 AWS 服务

列表。要选择目标服务以与 S3 on Outposts 结合使用，请参阅《Amazon CloudWatch Events 用户指南》中的使用 AWS CloudTrail 创建对 AWS API 调用触发的 CloudWatch Events 规则。

> **Note**
>
> 对于 S3 on Outposts 对象操作，只有将跟踪（可选带有事件选择器）配置为接收 CloudTrail 发送的 AWS API 调用事件时，这些事件才会匹配您的规则。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 CloudTrail 日志文件。

Example

以下是 `DeleteObject` 操作的示例规则。要使用此示例规则，请将 *amzn-s3-demo-bucket1* 替换为 S3 on Outposts 桶的名称。

```
{
  "source": [
    "aws.s3-outposts"
  ],
  "detail-type": [
    "AWS API call through CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3-outposts.amazonaws.com"
    ],
    "eventName": [
      "DeleteObject"
    ],
    "requestParameters": {
      "bucketName": [
        "amzn-s3-demo-bucket1"
      ]
    }
  }
}
```

使用 AWS CloudTrail 日志监控 S3 on Outposts

Amazon S3 on Outposts 与 AWS CloudTrail 集成，后者是在 S3 on Outposts 中提供用户、角色或 AWS 服务所采取操作的记录的服务。您可以使用 AWS CloudTrail 获取关于 S3 on Outposts 桶级别

请求以及对象级别请求的信息，这些请求旨在审计和记录您的 S3 on Outposts 事件活动。要为所有 Outposts 桶或特定 Outposts 桶的列表启用 CloudTrail 数据事件，您必须在 CloudTrail 中手动创建跟踪。有关 CloudTrail 日志文件条目的更多信息，请参阅 S3 on Outposts 日志文件条目。

> **Note**
>
> - 最佳实践是为 AWS CloudTrail 数据事件 Outposts 桶创建生命周期策略。配置生命周期策略，以便在您需要审计日志文件的时间段之后定期删除这些日志文件。这样做可以减少 Amazon Athena 为每个查询分析的数据量。有关更多信息，请参阅 在存储桶上设置生命周期配置。
> - 有关如何查询 CloudTrail 日志的示例，请参阅 AWS 大数据博客文章使用 AWS CloudTrail 和 Amazon Athena 分析安全性、合规性和操作活动。

为 S3 on Outposts 桶中的对象启用 CloudTrail 日志记录

您可以使用 Amazon S3 控制台配置 AWS CloudTrail 跟踪来记录 Amazon S3 on Outposts 桶中对象的数据事件。CloudTrail 支持记录 S3 on Outposts 对象级别 API 操作，例如 `GetObject`、`DeleteObject` 和 `PutObject`。这些事件称为数据事件。

原定设置情况下，CloudTrail 跟踪不记录数据事件。但是，您可以将跟踪配置为记录您指定的 S3 on Outposts 桶的数据事件，或记录 AWS 账户中所有 S3 on Outposts 桶的数据事件。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon S3 API 调用。

CloudTrail 不会在 CloudTrail 事件历史记录中填充数据事件。此外，并非所有 S3 on Outposts 桶级别 API 操作都会填充在 CloudTrail 事件历史记录中。有关如何查询 CloudTrail 日志的更多信息，请参阅 AWS 知识中心上的使用 Amazon CloudWatch Logs 筛选条件模式和 Amazon Athena 查询 CloudTrail 日志。

要配置跟踪以记录某个 S3 on Outposts 桶的数据事件，您可以使用 AWS CloudTrail 控制台或 Amazon S3 控制台。如果您要配置跟踪以记录您的 AWS 账户中所有 S3 on Outposts 桶的数据事件，使用 CloudTrail 控制台会更轻松。有关使用 CloudTrail 控制台配置跟踪以记录 S3 on Outposts 数据事件的信息，请参阅《AWS CloudTrail 用户指南》中的数据事件。

> **Important**
>
> 记录数据事件将收取额外费用。有关更多信息，请参阅 AWS CloudTrail 定价。

以下过程演示如何使用 Amazon S3 控制台配置 CloudTrail 跟踪来记录 S3 on Outposts 桶的数据事件。

> **Note**
>
> 创建桶的 AWS 账户拥有该桶，并且是唯一可以配置要发送到 AWS CloudTrail 的 S3 on Outposts 数据事件的账户。

为 S3 on Outposts 桶中的对象启用 CloudTrail 数据事件日志记录

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择 Outposts buckets（Outposts 桶）。
3. 选择要使用 CloudTrail 记录其数据事件的 Outposts 桶的名称。
4. 选择属性。
5. 在 AWS CloudTrail 数据事件部分中，选择在 CloudTrail 中配置。

   AWS CloudTrail 控制台将打开。

   您可以创建新的 CloudTrail 跟踪或重用现有跟踪，并配置要在跟踪中记录的 S3 on Outposts 数据事件。
6. 在 CloudTrail 控制台的控制面板页面上，选择创建跟踪。
7. 在步骤 1 选择跟踪属性页面上，提供跟踪的名称，选择用于存储跟踪日志的 S3 桶，指定所需的任何其他设置，然后选择下一步。
8. 在步骤 2 选择日志事件页面的事件类型下，选择数据事件。

   对于数据事件类型，选择 S3 Outposts。选择下一步。

   > **Note**
   >
   > - 当为 S3 on Outposts 创建跟踪和配置数据事件日志记录时，必须正确指定数据事件类型。
   >   - 如果您使用 CloudTrail 控制台，请对于数据事件类型选择 S3 Outposts。有关如何在 CloudTrail 控制台中创建跟踪的信息，请参阅 AWS CloudTrail 用户指南中的使用控制台创建和更新跟踪。有关如何在 CloudTrail 控制台中配置 S3 on Outposts 数据事件

日志记录的信息，请参阅《AWS CloudTrail 用户指南》中的记录 Amazon S3 对象的数据事件。

- 如果您使用 AWS Command Line Interface（AWS CLI）或 AWS SDK，请将 `resources.type` 字段设置为 `AWS::S3Outposts::Object`。有关如何使用 AWS CLI 记录 S3 on Outposts 数据事件的更多信息，请参阅《AWS CloudTrail 用户指南》中的记录 S3 on Outposts 事件。

- 如果您使用 CloudTrail 控制台或 Amazon S3 控制台配置某个跟踪以记录 S3 on Outposts 桶的数据事件，Amazon S3 控制台将显示已为该桶启用对象级别日志记录。

9. 在步骤 3 查看和创建页面上，查看您配置的跟踪属性和日志事件。然后，选择创建跟踪。

为 S3 on Outposts 桶中的对象禁用 CloudTrail 数据事件日志记录

1. 登录到 AWS Management Console，然后通过以下网址打开 CloudTrail 控制台：https://console.aws.amazon.com/cloudtrail。
2. 在左侧导航窗格中，选择跟踪。
3. 选择您创建用于记录 S3 on Outposts 桶的事件的跟踪名称。
4. 在跟踪的详细信息页面上，选择右上角的停止记录。
5. 在随后显示的对话框中，选择停止记录。

使用 Amazon S3 on Outposts 进行开发

通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 上创建 S3 桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松存储和检索对象。S3 on Outposts 提供了一个新的存储类 S3 Outposts (OUTPOSTS)；该存储类使用 Amazon S3 API，并且用于在 AWS Outposts 上的多个设备和服务器之间持久冗余地存储数据。您通过 Virtual Private Cloud（VPC）使用接入点和端点连接与 Outposts 桶进行通信。您可以像在 Amazon S3 桶中一样在 Outpost 桶上使用相同的 API 和功能，包括访问策略、加密和标记。您可以通过 AWS Management Console、AWS Command Line Interface (AWS CLI)、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅什么是 Amazon S3 on Outposts？

以下主题提供有关使用 S3 on Outposts 进行开发的信息。

主题

- Amazon S3 on Outposts API 操作

- 使用适用于 Java 的 SDK 为 S3 on Outposts 配置 S3 控制客户端
- 通过 IPv6 向 S3 on Outposts 发出请求

Amazon S3 on Outposts API 操作

本主题列出了您可以与 Amazon S3 on Outposts 一起使用的 Amazon S3、Amazon S3 Control 和 Amazon S3 on Outposts API 操作。

主题

- 用于管理对象的 Amazon S3 API 操作
- 用于管理存储桶的 Amazon S3 控制 API
- 用于管理端点的 S3 on Outposts API 操作

用于管理对象的 Amazon S3 API 操作

S3 on Outposts 旨在使用与 Amazon S3 相同的对象 API 操作。您必须使用访问点才能访问 Outpost 存储桶中的任何对象。将对象 API 操作与 S3 on Outposts 结合使用时，您需要提供 Outposts 访问点 Amazon 资源名称（ARN）或访问点别名。有关访问点别名的更多信息，请参阅为您的 S3 on Outposts 桶访问点使用桶式别名。

Amazon S3 on Outposts 支持以下Amazon S3 API 操作：

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- DeleteObjectTagging
- GetObject
- GetObjectTagging
- HeadBucket
- HeadObject
- ListMultipartUploads

- ListObjects
- ListObjectsV2
- ListObjectVersions
- ListParts
- PutObject
- PutObjectTagging
- UploadPart
- UploadPartCopy

用于管理存储桶的 Amazon S3 控制 API

S3 on Outposts 支持以下用于处理存储桶的 Amazon S3 控制 API 操作。

- CreateAccessPoint
- CreateBucket
- DeleteAccessPoint
- DeleteAccessPointPolicy
- DeleteBucket
- DeleteBucketLifecycleConfiguration
- DeleteBucketPolicy
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccessPoint
- GetAccessPointPolicy
- GetBucket
- GetBucketLifecycleConfiguration
- GetBucketPolicy
- GetBucketReplication
- GetBucketTagging
- GetBucketVersioning
- ListAccessPoints
- ListRegionalBuckets

- PutAccessPointPolicy
- PutBucketLifecycleConfiguration
- PutBucketPolicy
- PutBucketReplication
- PutBucketTagging
- PutBucketVersioning

用于管理端点的 S3 on Outposts API 操作

S3 on Outposts 支持以下用于管理终端节点的 Amazon S3 on Outposts API 操作。

- CreateEndpoint
- DeleteEndpoint
- ListEndpoints
- ListOutpostsWithS3
- ListSharedEndpoints

使用适用于 Java 的 SDK 为 S3 on Outposts 配置 S3 控制客户端

以下示例使用 AWS SDK for Java 为 Amazon S3 on Outposts 配置 Amazon S3 控制客户端。要使用此示例，请将每个 *user input placeholder* 替换为您自己的信息。

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;

public AWSS3Control createS3ControlClient() {

    String accessKey = AWSAccessKey;
    String secretKey = SecretAccessKey;
    BasicAWSCredentials awsCreds = new BasicAWSCredentials(accessKey, secretKey);

    return AWSS3ControlClient.builder().enableUseArnRegion()
            .withCredentials(new AWSStaticCredentialsProvider(awsCreds))
            .build();
```

```
}
```

通过 IPv6 向 S3 on Outposts 发出请求

Amazon S3 on Outposts 和 S3 on Outposts 双堆栈端点支持使用 IPv6 或 IPv4 协议向 S3 on Outposts 桶发出的请求。借助 S3 on Outposts 对 IPv6 的支持，可以通过 IPv6 网络使用 S3 on Outposts API 访问和操作桶以及控制面板资源。

> **Note**
>
> 不支持 IPv6 网络上的 S3 on Outposts 对象操作（例如 `PutObject` 或 `GetObject`）。

通过 IPv6 网络访问 S3 on Outposts 不额外收费。有关 S3 on Outposts 的更多信息，请参阅 S3 on Outposts 定价。

主题

- IPv6 入门
- 使用双堆栈端点通过 IPv6 网络发出请求
- 在 IAM 策略中使用 IPv6 地址
- 测试 IP 地址兼容性
- 将 IPv6 与 AWS PrivateLink 结合使用
- 使用 S3 on Outposts 双堆栈端点

IPv6 入门

要通过 IPv6 向 S3 on Outposts 桶发出请求，您必须使用双堆栈端点。下一节介绍如何使用双堆栈终端节点通过 IPv6 发出请求。

下面是在尝试通过 IPv6 访问 S3 on Outposts 桶之前的重要注意事项：

- 访问存储桶的客户端和网络必须支持使用 IPv6。
- 虚拟托管类型和路径类型请求必须都受支持，以便进行 IPv6 访问。有关更多信息，请参阅使用 S3 on Outposts 双堆栈端点。
- 如果您在 AWS Identity and Access Management（IAM）用户策略或 S3 on Outposts 桶策略中使用源 IP 地址筛选，则必须更新策略以包括 IPv6 地址范围。

> **Note**
>
> 此要求仅适用于跨 IPv6 网络的 S3 on Outposts 桶操作和控制面板资源。不支持跨 IPv6 网络的 Amazon S3 on Outposts 对象操作。

- 如果使用 IPv6，服务器访问日志文件以 IPv6 格式输出 IP 地址。您必须对用于分析 S3 on Outposts 日志文件的现有工具、脚本和软件进行更新，以便它们能够分析 IPv6 格式的远程 IP 地址。然后，更新的工具、脚本和软件将正确解析 IPv6 格式的远程 IP 地址。

使用双堆栈端点通过 IPv6 网络发出请求

要通过 IPv6 使用 S3 on Outposts API 调用发出请求，可以通过 AWS CLI 或 AWS SDK 使用双堆栈端点。无论您是通过 IPv6 协议还是 IPv4 协议访问 S3 on Outposts，Amazon S3 控制 API 操作和 S3 on Outposts API 操作的工作方式都是相同的。但请注意，IPv6 网络不支持 S3 on Outposts 对象操作（例如 `PutObject` 或 `GetObject`）。

如果使用 AWS Command Line Interface（AWS CLI）和 AWS SDK，可使用参数或标志以更改为双堆栈端点。您还可以在配置文件中直接将双堆栈端点指定为覆盖 S3 on Outposts 端点。

您可以通过以下任一方式，使用双堆栈端点通过 IPv6 访问 S3 on Outposts 桶：

- AWS CLI，请参阅从 AWS CLI 使用双堆栈端点。
- 有关 AWS 开发工具包，请参阅 从 AWS SDK 使用 S3 on Outposts 双堆栈端点。

在 IAM 策略中使用 IPv6 地址

在尝试使用 IPv6 协议访问 S3 on Outposts 桶之前，请确保用于 IP 地址筛选的 IAM 用户策略或 S3 on Outposts 桶策略已更新为包括 IPv6 地址范围。如果未更新 IP 地址筛选策略以便处理 IPv6 地址，则在尝试使用 IPv6 协议时，您可能会无法访问 S3 on Outposts 桶。

筛选 IP 地址的 IAM policy 使用 IP 地址条件运算符。下面的 S3 on Outposts 桶策略通过使用 IP 地址条件运算符，确定允许的 IPv4 地址的 IP 范围为 54.240.143.*。此范围之外的所有 IP 地址对 S3 on Outposts 桶（`DOC-EXAMPLE-BUCKET`）的访问都会被拒绝。由于所有 IPv6 地址都在允许范围之外，此策略会阻止 IPv6 地址访问 `DOC-EXAMPLE-BUCKET`。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3outposts:*",
      "Resource": "arn:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/DOC-EXAMPLE-BUCKET/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

您可以将 S3 on Outposts 桶策略的 `Condition` 元素修改为同时允许 IPv4（`54.240.143.0/24`）和 IPv6（`2001:DB8:1234:5678::/64`）地址范围，如以下示例所示。您可以使用示例中所示的相同类型的 `Condition` 块来更新 IAM 用户和存储桶策略。

```
       "Condition": {
         "IpAddress": {
            "aws:SourceIp": [
              "54.240.143.0/24",
               "2001:DB8:1234:5678::/64"
             ]
          }
        }
```

在使用 IPv6 之前，您必须对使用 IP 地址筛选来允许 IPv6 地址范围的所有相关 IAM 用户和存储桶策略进行更新。建议您使用组织的 IPv6 地址范围以及现有的 IPv4 地址范围来更新您的 IAM 策略。有关同时允许通过 IPv6 和 IPv4 进行访问的存储桶策略示例，请参阅限制特定 IP 地址的访问权限。

您可以在 https://console.aws.amazon.com/iam/ 上使用 IAM 控制台查看 IAM 用户策略。有关 IAM 的更多信息，请参阅 IAM 用户指南。有关编辑 S3 on Outposts 桶策略的信息，请参阅为 Amazon S3 on Outposts 存储桶添加或编辑存储桶策略。

测试 IP 地址兼容性

如果您使用的是 Linux 或 Unix 实例或 macOS X 平台，则可以测试您通过 IPv6 对双堆栈端点的访问。例如，要测试通过 IPv6 与 Amazon S3 on Outposts 端点的连接，请使用 `dig` 命令：

```
dig s3-outposts.us-west-2.api.aws AAAA +short
```

如果正确设置了通过 IPv6 网络的双堆栈端点，则 dig 命令将返回连接的 IPv6 地址。例如：

```
dig s3-outposts.us-west-2.api.aws AAAA +short

2600:1f14:2588:4800:b3a9:1460:159f:ebce

2600:1f14:2588:4802:6df6:c1fd:ef8a:fc76

2600:1f14:2588:4801:d802:8ccf:4e04:817
```

将 IPv6 与 AWS PrivateLink 结合使用

S3 on Outposts 支持对 AWS PrivateLink 服务和端点使用 IPv6 协议。由于 AWS PrivateLink 支持 IPv6 协议，您可以通过 IPv6 网络，从本地或通过其它私有连接来连接到 VPC 内的服务端点。AWS PrivateLink for S3 on Outposts 对 IPv6 的支持还允许您将 AWS PrivateLink 与双堆栈端点集成。有关如何为 AWS PrivateLink 启用 IPv6 的步骤，请参阅 Expedite your IPv6 adoption with AWS PrivateLink services and endpoints。

> **Note**
>
> 要将支持的 IP 地址类型从 IPv4 更新为 IPv6，请参阅《AWS PrivateLink User Guide》中的 Modify the supported IP address type。

将 IPv6 与 AWS PrivateLink 结合使用

如果您将 AWS PrivateLink 与 IPv6 结合使用，则必须创建 IPv6 或双堆栈 VPC 接口端点。有关如何使用 AWS Management Console创建 VPC 端点的一般步骤，请参阅《AWS PrivateLink User Guide》中的 Access an AWS service using an interface VPC endpoint。

AWS Management Console

使用以下过程创建连接到 S3 on Outposts 的接口 VPC 端点。

1. 登录到 AWS Management Console并打开 VPC 控制台，网址为：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择端点。
3. 选择 创建端点。
4. 对于服务类别，选择 AWS 服务。

5. 对于服务名称，选择 S3 on Outposts 服务（com.amazonaws.us-east-1.s3-outposts）。
6. 对于 VPC，选择您要从中访问 S3 on Outposts 的 VPC。
7. 对于子网，对于每个可用区选择一个您将从中访问 S3 on Outposts 的子网。您无法从同一可用区中选择多个子网。对于您选择的每个子网，将创建一个新的端点网络接口。默认情况下，将子网 IP 地址范围中的 IP 地址分配给端点网络接口。要为端点网络接口指定 IP 地址，请选择指定 IP 地址，然后输入子网地址范围中的 IPv6 地址。
8. 对于 IP 地址类型，选择双堆栈。同时将 IPv4 和 IPv6 地址分配给端点网络接口。仅当所有选定子网都具有 IPv4 和 IPv6 地址范围时，才支持此选项。
9. 对于安全组，选择要与 VPC 端点的端点网络接口关联的安全组。默认情况下，默认安全组与 VPC 相关联。
10. 对于策略，请选择完整访问权限以允许所有主体通过 VPC 端点对所有资源执行所有操作。否则，选择自定义来附加 VPC 端点策略，该策略控制主体通过 VPC 端点对资源执行操作的权限。该选项仅在服务支持 VPC 端点策略时可用。有关更多信息，请参阅 Endpoint policies。
11. （可选）若要添加标签，请选择 Add new tag（添加新标签），然后输入该标签的键和值。
12. 选择创建端点。

Example – S3 on Outposts 桶策略

要允许 S3 on Outposts 与 VPC 端点进行交互，您可以更新您的 S3 on Outposts 策略，如下所示：

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3-outposts:*",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

AWS CLI

> **Note**
>
> 要在您的 VPC 端点上启用 IPv6 网络，必须为 S3 on Outposts 的 `SupportedIpAddressType` 筛选条件设置 `IPv6`。

以下示例使用 `create-vpc-endpoint` 命令创建新的双堆栈接口端点。

```
aws ec2 create-vpc-endpoint \
--vpc-id vpc-12345678 \
--vpc-endpoint-type Interface \
--service-name com.amazonaws.us-east-1.s3-outposts \
--subnet-id subnet-12345678 \
--security-group-id sg-12345678 \
--ip-address-type dualstack \
--dns-options "DnsRecordIpType=dualstack"
```

根据 AWS PrivateLink 服务配置，新创建的端点连接可能需要先由 VPC 端点服务提供商接受，然后才能使用。有关更多信息，请参阅《AWS PrivateLink User Guide》中的 Accept and reject endpoint connection requests。

以下示例使用 `modify-vpc-endpoint` 命令将仅限 IPv4 的 VPC 端点更新为双堆栈端点。双堆栈端点同时允许访问 IPv4 和 IPv6 网络。

```
aws ec2 modify-vpc-endpoint \
--vpc-endpoint-id vpce-12345678 \
--add-subnet-ids subnet-12345678 \
--remove-subnet-ids subnet-12345678 \
--ip-address-type dualstack \
--dns-options "DnsRecordIpType=dualstack"
```

有关如何为 AWS PrivateLink 启用 IPv6 网络的更多信息，请参阅 Expedite your IPv6 adoption with AWS PrivateLink services and endpoints。

使用 S3 on Outposts 双堆栈端点

S3 on Outposts 双堆栈端点支持通过 IPv6 和 IPv4 向 S3 on Outposts 桶发出的请求。本节介绍如何使用 S3 on Outposts 双堆栈端点。

主题

- S3 on Outposts 双堆栈端点
- 从 AWS CLI 使用双堆栈端点
- 从 AWS SDK 使用 S3 on Outposts 双堆栈端点

S3 on Outposts 双堆栈端点

当您向双堆栈端点发出请求时，S3 on Outposts 桶 URL 解析为 IPv6 或 IPv4 地址。有关如何通过 IPv6 访问 S3 on Outposts 桶的更多信息，请参阅通过 IPv6 向 S3 on Outposts 发出请求。

要通过双堆栈端点访问 S3 on Outposts 桶，请使用路径样式端点名称。S3 on Outposts 只支持区域双堆栈端点名称，这意味着，您必须在名称中指定区域。

对于双堆栈路径样式的 FIP 端点，请使用以下命名约定：

```
s3-outposts-fips.region.api.aws
```

对于双堆栈非 FIPS 端点，请使用以下命名约定：

```
s3-outposts.region.api.aws
```

> **Note**
>
> S3 on Outposts 不支持虚拟托管样式的端点名称。

从 AWS CLI 使用双堆栈端点

本节提供用于向双堆栈端点发出请求的 AWS CLI 命令示例。有关设置 AWS CLI 的说明，请参阅通过 AWS CLI 和适用于 Java 的 SDK 开始使用。

在 AWS Config 文件内的配置文件中将配置值 `use_dualstack_endpoint` 设置为 `true`，从而将 `s3` 和 `s3api` AWS CLI 命令发出的所有 Amazon S3 请求都定向到指定区域的双堆栈端点。您可以在配置文件或命令中使用 `--region` 选项指定区域。

通过 AWS CLI 使用双堆栈端点时，仅支持 `path` 寻址样式。在配置文件中设置的寻址样式确定桶名称是在主机名中还是在 URL 中。有关更多信息，请参阅《AWS CLI 用户指南》中的 s3outposts。

要通过 AWS CLI 使用双栈端点，请将 `--endpoint-url` 参数与 `http://s3.dualstack.region.amazonaws.com` 或 `https://s3-outposts-fips.region.api.aws` 端点一起用于任何 `s3control` 或 `s3outposts` 命令。

例如：

```
$  aws s3control list-regional-buckets --endpoint-url https://s3-
outposts.region.api.aws
```

从 AWS SDK 使用 S3 on Outposts 双堆栈端点

本节提供一些示例，介绍如何使用 AWS SDK 来访问双堆栈端点。

AWS SDK for Java 2.x双堆栈端点示例

以下示例显示了当使用 AWS SDK for Java 2.x 创建 S3 on Outposts 客户端时，如何使用 `S3ControlClient` 和 `S3OutpostsClient` 类来启用双堆栈端点。有关为 Amazon S3 on Outposts 创建和测试有效 Java 示例的说明，请参阅通过 AWS CLI 和适用于 Java 的 SDK 开始使用。

Example – 创建启用双堆栈端点的 **`S3ControlClient`** 类

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsRequest;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsResponse;
import software.amazon.awssdk.services.s3control.model.S3ControlException;


public class DualStackEndpointsExample1 {

    public static void main(String[] args) {
        Region clientRegion = Region.of("us-east-1");
        String accountId = "111122223333";
        String navyId = "9876543210";

        try {
            // Create an S3ControlClient with dual-stack endpoints enabled.
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                                                             .region(clientRegion)
                                                             .dualstackEnabled(true)
                                                             .build();
            ListRegionalBucketsRequest listRegionalBucketsRequest =
 ListRegionalBucketsRequest.builder()

       .accountId(accountId)

       .outpostId(navyId)

       .build();
```

```
            ListRegionalBucketsResponse listBuckets =
 s3ControlClient.listRegionalBuckets(listRegionalBucketsRequest);
            System.out.printf("ListRegionalBuckets Response: %s%n",
 listBuckets.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        }
        catch (S3ControlException e) {
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

Example – 创建启用双堆栈端点的 **S3OutpostsClient**

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3outposts.S3OutpostsClient;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsRequest;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsResponse;
import software.amazon.awssdk.services.s3outposts.model.S3OutpostsException;


public class DualStackEndpointsExample2 {

    public static void main(String[] args) {
        Region clientRegion = Region.of("us-east-1");

        try {
            // Create an S3OutpostsClient with dual-stack endpoints enabled.
            S3OutpostsClient s3OutpostsClient = S3OutpostsClient.builder()
                                                                .region(clientRegion)
                                                                .dualstackEnabled(true)
```

```
                                                                .build();
            ListEndpointsRequest listEndpointsRequest =
 ListEndpointsRequest.builder().build();

            ListEndpointsResponse listEndpoints =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
            System.out.printf("ListEndpoints Response: %s%n",
 listEndpoints.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        }
        catch (S3OutpostsException e) {
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

如果要在 Windows 上使用 AWS SDK for Java 2.x，可能必须设置以下 Java 虚拟机（JVM）属性：

```
java.net.preferIPv6Addresses=true
```

适用于使用 AWS 软件开发工具包的 Amazon S3 的代码示例

以下代码示例演示了如何将 Amazon S3 与 AWS 软件开发工具包 (SDK) 一起使用。

基础知识是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务 结合来完成特定任务的代码示例。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用

Hello Amazon S3

以下代码示例演示了如何开始使用 Amazon S3。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
```

```
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_s3.cpp 源文件的代码。

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
```

```
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 *  A "Hello S3" starter application which initializes an Amazon Simple Storage
 Service (Amazon S3) client
 *  and lists the Amazon S3 buckets in the selected region.
 *
 *  main function
 *
 *  Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the
 S3 service permits anonymous requests, thus the s3Client will return "success"
 and 0 buckets even if you are unauthenticated, which can be confusing to a new
 user.
        auto provider =
 Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed with error: " << outcome.GetError() <<
 std::endl;
            result = 1;
```

```
        } else {
            std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                      << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 ListBuckets。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
```

```
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 ListBuckets。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 ListBuckets。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";

// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new S3Client({});

export const helloS3 = async () => {
  const command = new ListBucketsCommand({});

  const { Buckets } = await client.send(command);
  console.log("Buckets: ");
  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
  return Buckets;
};
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 ListBuckets。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
use Aws\S3\S3Client;
```

```
$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- 有关 API 详细信息，请参阅《AWS SDK for PHP API 参考》中的 ListBuckets。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import boto3


def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) resource and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    s3_resource = boto3.resource("s3")
    print("Hello, Amazon S3! Let's list your buckets:")
    for bucket in s3_resource.buckets.all():
        print(f"\t{bucket.name}")


if __name__ == "__main__":
    hello_s3()
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 ListBuckets。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
```

```
  manager.list_buckets
end
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 ListBuckets。

代码示例

- 使用 AWS SDK 的 Amazon S3 的基本示例
  - Hello Amazon S3
  - 了解使用 AWS SDK 的 Amazon S3 的基本功能
  - 使用 AWS SDK 对 Amazon S3 执行的操作
    - 将 AbortMultipartUpload 与 AWS SDK 或 CLI 配合使用
    - 将 AbortMultipartUploads 与 AWS SDK 或 CLI 配合使用
    - 将 CompleteMultipartUpload 与 AWS SDK 或 CLI 配合使用
    - 将 CopyObject 与 AWS SDK 或 CLI 配合使用
    - 将 CreateBucket 与 AWS SDK 或 CLI 配合使用
    - 将 CreateMultiRegionAccessPoint 与 AWS SDK 或 CLI 配合使用
    - 将 CreateMultipartUpload 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucket 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketAnalyticsConfiguration 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketCors 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketEncryption 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketInventoryConfiguration 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketLifecycle 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketMetricsConfiguration 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketPolicy 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketReplication 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketTagging 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteBucketWebsite 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteObject 与 AWS SDK 或 CLI 配合使用
    - 将 DeleteObjectTagging 与 AWS SDK 或 CLI 配合使用

- 将 DeleteObjects 与 AWS SDK 或 CLI 配合使用
- 将 DeletePublicAccessBlock 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketAccelerateConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketAcl 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketAnalyticsConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketCors 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketEncryption 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketInventoryConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketLocation 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketLogging 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketMetricsConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketNotification 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketPolicy 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketPolicyStatus 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketReplication 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketRequestPayment 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketTagging 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketVersioning 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketWebsite 与 AWS SDK 或 CLI 配合使用
- 将 GetObject 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectAcl 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectAttributes 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectLegalHold 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectRetention 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectTagging 与 AWS SDK 或 CLI 配合使用
- 将 GetPublicAccessBlock 与 AWS SDK 或 CLI 配合使用
- 将 HeadBucket 与 AWS SDK 或 CLI 配合使用
- 将 HeadObject 与 AWS SDK 或 CLI 配合使用

- 将 ListBucketAnalyticsConfigurations 与 AWS SDK 或 CLI 配合使用
- 将 ListBucketInventoryConfigurations 与 AWS SDK 或 CLI 配合使用
- 将 ListBuckets 与 AWS SDK 或 CLI 配合使用
- 将 ListMultipartUploads 与 AWS SDK 或 CLI 配合使用
- 将 ListObjectVersions 与 AWS SDK 或 CLI 配合使用
- 将 ListObjects 与 AWS SDK 或 CLI 配合使用
- 将 ListObjectsV2 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketAccelerateConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketAcl 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketCors 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketEncryption 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketLogging 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketNotification 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketNotificationConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketPolicy 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketReplication 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketRequestPayment 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketTagging 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketVersioning 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketWebsite 与 AWS SDK 或 CLI 配合使用
- 将 PutObject 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectAcl 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectLegalHold 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectRetention 与 AWS SDK 或 CLI 配合使用
- 将 RestoreObject 与 AWS SDK 或 CLI 配合使用
- 将 SelectObjectContent 与 AWS SDK 或 CLI 配合使用
- 将 UploadPart 与 AWS SDK 或 CLI 配合使用

- 适用于使用 AWS 软件开发工具包的 Amazon S3 的场景

- 使用 AWS SDK 将文本转换为语音以及将语音转换回文本
- 使用 AWS SDK 创建 Amazon S3 的预签名 URL
- 创建照片资产管理应用程序，让用户能够使用标签管理照片
- 使用 AWS SDK 列出 Amazon S3 对象的网页
- 创建 Amazon Textract 浏览器应用程序
- 使用 AWS SDK 删除未完成的 Amazon S3 分段上传
- 使用 AWS SDK 通过 Amazon Rekognition 检测图像中的 PPE
- 使用 AWS SDK 检测从图像中提取的文本中的实体
- 使用 AWS SDK 检测图像中的人脸
- 使用 AWS 软件开发工具包通过 Amazon Rekognition 检测图像中的对象
- 使用 AWS SDK 通过 Amazon Rekognition 检测视频中的人物和对象
- 将 Amazon Simple Storage Service（Amazon S3）桶中的所有对象下载到本地目录
- 使用 AWS SDK 从多区域接入点中获取 Amazon S3 对象
- 使用 AWS SDK 并指定 If-Modified-Since 标头以从 Amazon S3 存储桶获取对象
- 使用 AWS SDK 加密 Amazon S3 对象入门
- 使用 AWS SDK 为 Amazon S3 存储桶和对象加标签入门
- 使用 AWS SDK 获取 Amazon S3 存储桶的法定保留配置
- 通过 AWS SDK 使用 Amazon S3 对象锁定功能
- 使用 AWS SDK 管理 Amazon S3 存储桶的访问控制列表（ACL）
- 通过 AWS SDK 使用 Lambda 函数批量管理版本控制的 Amazon S3 对象
- 使用 AWS SDK 解析 Amazon S3 URI
- 使用 AWS SDK 执行 Amazon S3 对象的分段复制
- 使用 AWS SDK 执行 Amazon S3 对象的分段上传
- 使用 AWS SDK 接收和处理 Amazon S3 事件通知。
- 使用 AWS SDK 保存 EXIF 和其他图像信息
- 使用 AWS SDK 向 Amazon EventBridge 发送 S3 事件通知
- 跟踪使用 AWS SDK 执行的 Amazon S3 对象上传或下载操作
- 使用 S3 对象 Lambda 转换应用程序的数据
- 使用 AWS SDK 进行单元测试和集成测试的示例方法

- 以递归方式将本地目录上传到 Amazon Simple Storage Service（Amazon S3）桶

- 借助 AWS SDK 向 Amazon S3 上传大文件或从 Amazon S3 下载大文件
- 使用 AWS SDK 将未知大小的流上传到 Amazon S3 对象
- 使用 AWS 开发工具包，对 Amazon S3 对象使用校验和
- 通过 AWS SDK 使用 Amazon S3 对象完整性功能
- 使用 AWS SDK 使用 Amazon S3 版本控制对象
- 使用 AWS 开发工具包的 Amazon S3 无服务器示例
  - 通过 Amazon S3 触发器调用 Lambda 函数

使用 AWS SDK 的 Amazon S3 的基本示例

以下代码示例展示了如何将 Amazon Simple Storage Service 的基本功能与 AWS SDK 结合使用。

示例

- Hello Amazon S3
- 了解使用 AWS SDK 的 Amazon S3 的基本功能
- 使用 AWS SDK 对 Amazon S3 执行的操作
  - 将 AbortMultipartUpload 与 AWS SDK 或 CLI 配合使用
  - 将 AbortMultipartUploads 与 AWS SDK 或 CLI 配合使用
  - 将 CompleteMultipartUpload 与 AWS SDK 或 CLI 配合使用
  - 将 CopyObject 与 AWS SDK 或 CLI 配合使用
  - 将 CreateBucket 与 AWS SDK 或 CLI 配合使用
  - 将 CreateMultiRegionAccessPoint 与 AWS SDK 或 CLI 配合使用
  - 将 CreateMultipartUpload 与 AWS SDK 或 CLI 配合使用
  - 将 DeleteBucket 与 AWS SDK 或 CLI 配合使用
  - 将 DeleteBucketAnalyticsConfiguration 与 AWS SDK 或 CLI 配合使用
  - 将 DeleteBucketCors 与 AWS SDK 或 CLI 配合使用
  - 将 DeleteBucketEncryption 与 AWS SDK 或 CLI 配合使用
  - 将 DeleteBucketInventoryConfiguration 与 AWS SDK 或 CLI 配合使用
  - 将 DeleteBucketLifecycle 与 AWS SDK 或 CLI 配合使用
  - 将 DeleteBucketMetricsConfiguration 与 AWS SDK 或 CLI 配合使用
  - 将 DeleteBucketPolicy 与 AWS SDK 或 CLI 配合使用

- 将 DeleteBucketReplication 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketTagging 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketWebsite 与 AWS SDK 或 CLI 配合使用
- 将 DeleteObject 与 AWS SDK 或 CLI 配合使用
- 将 DeleteObjectTagging 与 AWS SDK 或 CLI 配合使用
- 将 DeleteObjects 与 AWS SDK 或 CLI 配合使用
- 将 DeletePublicAccessBlock 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketAccelerateConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketAcl 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketAnalyticsConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketCors 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketEncryption 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketInventoryConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketLocation 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketLogging 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketMetricsConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketNotification 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketPolicy 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketPolicyStatus 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketReplication 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketRequestPayment 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketTagging 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketVersioning 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketWebsite 与 AWS SDK 或 CLI 配合使用
- 将 GetObject 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectAcl 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectAttributes 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectLegalHold 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用

- 将 GetObjectRetention 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectTagging 与 AWS SDK 或 CLI 配合使用
- 将 GetPublicAccessBlock 与 AWS SDK 或 CLI 配合使用
- 将 HeadBucket 与 AWS SDK 或 CLI 配合使用
- 将 HeadObject 与 AWS SDK 或 CLI 配合使用
- 将 ListBucketAnalyticsConfigurations 与 AWS SDK 或 CLI 配合使用
- 将 ListBucketInventoryConfigurations 与 AWS SDK 或 CLI 配合使用
- 将 ListBuckets 与 AWS SDK 或 CLI 配合使用
- 将 ListMultipartUploads 与 AWS SDK 或 CLI 配合使用
- 将 ListObjectVersions 与 AWS SDK 或 CLI 配合使用
- 将 ListObjects 与 AWS SDK 或 CLI 配合使用
- 将 ListObjectsV2 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketAccelerateConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketAcl 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketCors 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketEncryption 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketLogging 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketNotification 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketNotificationConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketPolicy 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketReplication 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketRequestPayment 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketTagging 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketVersioning 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketWebsite 与 AWS SDK 或 CLI 配合使用
- 将 PutObject 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectAcl 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectLegalHold 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用

- 将 PutObjectRetention 与 AWS SDK 或 CLI 配合使用
- 将 RestoreObject 与 AWS SDK 或 CLI 配合使用
- 将 SelectObjectContent 与 AWS SDK 或 CLI 配合使用
- 将 UploadPart 与 AWS SDK 或 CLI 配合使用

Hello Amazon S3

以下代码示例展示了如何开始使用 Amazon S3。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
```

```
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_s3.cpp 源文件的代码。

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 *  A "Hello S3" starter application which initializes an Amazon Simple Storage
 Service (Amazon S3) client
 *  and lists the Amazon S3 buckets in the selected region.
 *
 *  main function
 *
```

```
 *  Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the
 S3 service permits anonymous requests, thus the s3Client will return "success"
 and 0 buckets even if you are unauthenticated, which can be confusing to a new
 user.
        auto provider =
 Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed with error: " << outcome.GetError() <<
 std::endl;
            result = 1;
        } else {
            std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                      << " buckets\n";
            for (auto &bucket: outcome.GetResult().GetBuckets()) {
                std::cout << bucket.GetName() << std::endl;
            }
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
```

```
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 ListBuckets。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 s3Client := s3.NewFromConfig(sdkConfig)
 count := 10
 fmt.Printf("Let's list up to %v buckets for your account.\n", count)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
```

```
 if err != nil {
  fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Buckets) == 0 {
  fmt.Println("You don't have any buckets!")
 } else {
  if count > len(result.Buckets) {
   count = len(result.Buckets)
  }
  for _, bucket := range result.Buckets[:count] {
   fmt.Printf("\t%v\n", *bucket.Name)
  }
 }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 ListBuckets。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 ListBuckets。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";

// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new S3Client({});

export const helloS3 = async () => {
  const command = new ListBucketsCommand({});

  const { Buckets } = await client.send(command);
  console.log("Buckets: ");
  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
  return Buckets;
};
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 ListBuckets。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
use Aws\S3\S3Client;

$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- 有关 API 详细信息，请参阅《AWS SDK for PHP API 参考》中的 ListBuckets。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import boto3


def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) resource and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    s3_resource = boto3.resource("s3")
    print("Hello, Amazon S3! Let's list your buckets:")
    for bucket in s3_resource.buckets.all():
        print(f"\t{bucket.name}")


if __name__ == "__main__":
    hello_s3()
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 ListBuckets。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 ListBuckets。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

了解使用 AWS SDK 的 Amazon S3 的基本功能

以下代码示例演示了如何：

- 创建存储桶并将文件上传到其中。
- 从桶中下载对象。
- 将对象复制到存储桶中的子文件夹。
- 列出存储桶中的对象。
- 删除桶对象和桶。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
public class S3_Basics
{
    public static async Task Main()
    {
        // Create an Amazon S3 client object. The constructor uses the
        // default user installed on the system. To work with Amazon S3
        // features in a different AWS Region, pass the AWS Region as a
        // parameter to the client constructor.
        IAmazonS3 client = new AmazonS3Client();
        string bucketName = string.Empty;
        string filePath = string.Empty;
        string keyName = string.Empty;

        var sepBar = new string('-', Console.WindowWidth);

        Console.WriteLine(sepBar);
        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
        Console.WriteLine("procedures. This application will:");
        Console.WriteLine("\n\t1. Create a bucket");
        Console.WriteLine("\n\t2. Upload an object to the new bucket");
```

```
            Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
 bucket");
            Console.WriteLine("\n\t4. List the items in the new bucket");
            Console.WriteLine("\n\t5. Delete all the items in the bucket");
            Console.WriteLine("\n\t6. Delete the bucket");
            Console.WriteLine(sepBar);

            // Create a bucket.
            Console.WriteLine($"\n{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);

            Console.Write("Please enter a name for the new bucket: ");
            bucketName = Console.ReadLine();

            var success = await S3Bucket.CreateBucketAsync(client, bucketName);
            if (success)
            {
                Console.WriteLine($"Successfully created bucket: {bucketName}.
\n");
            }
            else
            {
                Console.WriteLine($"Could not create bucket: {bucketName}.\n");
            }

            Console.WriteLine(sepBar);
            Console.WriteLine("Upload a file to the new bucket.");
            Console.WriteLine(sepBar);

            // Get the local path and filename for the file to upload.
            while (string.IsNullOrEmpty(filePath))
            {
                Console.Write("Please enter the path and filename of the file to
 upload: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (!File.Exists(filePath))
                {
                    Console.WriteLine($"Couldn't find {filePath}. Try again.\n");
                    filePath = string.Empty;
                }
            }
```

```
            // Get the file name from the full path.
            keyName = Path.GetFileName(filePath);

            success = await S3Bucket.UploadFileAsync(client, bucketName, keyName,
filePath);

            if (success)
            {
                Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
            }
            else
            {
                Console.WriteLine($"Could not upload {keyName}.\n");
            }

            // Set the file path to an empty string to avoid overwriting the
            // file we just uploaded to the bucket.
            filePath = string.Empty;

            // Now get a new location where we can save the file.
            while (string.IsNullOrEmpty(filePath))
            {
                // First get the path to which the file will be downloaded.
                Console.Write("Please enter the path where the file will be
downloaded: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (File.Exists($"{filePath}\\{keyName}"))
                {
                    Console.WriteLine($"Sorry, the file already exists in that
location.\n");
                    filePath = string.Empty;
                }
            }

            // Download an object from a bucket.
            success = await S3Bucket.DownloadObjectFromBucketAsync(client,
bucketName, keyName, filePath);

            if (success)
            {
```

```
                Console.WriteLine($"Successfully downloaded {keyName}.\n");
            }
            else
            {
                Console.WriteLine($"Sorry, could not download {keyName}.\n");
            }

            // Copy the object to a different folder in the bucket.
            string folderName = string.Empty;

            while (string.IsNullOrEmpty(folderName))
            {
                Console.Write("Please enter the name of the folder to copy your
object to: ");
                folderName = Console.ReadLine();
            }

            while (string.IsNullOrEmpty(keyName))
            {
                // Get the name to give to the object once uploaded.
                Console.Write("Enter the name of the object to copy: ");
                keyName = Console.ReadLine();
            }

            await S3Bucket.CopyObjectInBucketAsync(client, bucketName, keyName,
folderName);

            // List the objects in the bucket.
            await S3Bucket.ListBucketContentsAsync(client, bucketName);

            // Delete the contents of the bucket.
            await S3Bucket.DeleteBucketContentsAsync(client, bucketName);

            // Deleting the bucket too quickly after deleting its contents will
            // cause an error that the bucket isn't empty. So...
            Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
            _ = Console.ReadLine();

            // Delete the bucket.
            await S3Bucket.DeleteBucketAsync(client, bucketName);
        }
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for .NET API 参考中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
###############################################################################
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function s3_getting_started() {
  {
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
      cd bucket-lifecycle-operations || exit

      source ./bucket_operations.sh
      cd ..
    fi
```

```
}

echo_repeat "*" 88
echo "Welcome to the Amazon S3 getting started demo."
echo_repeat "*" 88

local bucket_name
bucket_name=$(generate_random_name "doc-example-bucket")

local region_code
region_code=$(aws configure get region)

if create_bucket -b "$bucket_name" -r "$region_code"; then
  echo "Created demo bucket named $bucket_name"
else
  errecho "The bucket failed to create. This demo will exit."
  return 1
fi

local file_name
while [ -z "$file_name" ]; do
  echo -n "Enter a file you want to upload to your bucket: "
  get_input
  file_name=$get_input_result

  if [ ! -f "$file_name" ]; then
    echo "Could not find file $file_name. Are you sure it exists?"
    file_name=""
  fi
done

local key
key="$(basename "$file_name")"

local result=0
if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
  echo "Uploaded file $file_name into bucket $bucket_name with key $key."
else
  result=1
fi

local destination_file
destination_file="$file_name.download"
```

```
 if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
   if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
     echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
   else
     result=1
   fi
 fi

 if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
   local to_key
   to_key="demo/$key"
   if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
     echo "Copied $key in the bucket $bucket_name to the  $to_key."
   else
     result=1
   fi
 fi

 local bucket_items
 bucket_items=$(list_items_in_bucket "$bucket_name")

 # shellcheck disable=SC2181
 if [[ $? -ne 0 ]]; then
   result=1
 fi

 echo "Your bucket contains the following items."
 echo -e "Name\t\tSize"
 echo "$bucket_items"

 if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
(y/n) "; then
   bucket_items=$(echo "$bucket_items" | cut -f 1)

   if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
     echo "The following items were deleted from the bucket $bucket_name"
     echo "$bucket_items"
   else
     result=1
   fi
```

```
    if delete_bucket "$bucket_name"; then
      echo "Deleted the bucket $bucket_name"
    else
      result=1
    fi
  fi

  return $result
}
```

此场景中使用的 Amazon S3 函数。

```
###############################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
```

```
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi
```

```
  # shellcheck disable=SC2086
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

```
################################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
###############################################################################
function copy_item_in_bucket() {
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}

###############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')
```

```
  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}

################################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
```

```
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}

################################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.

# Returns:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2

- PutObject

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param client: An S3 client.
          \return bool: Function succeeded.
        */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);
```

```
        //! Delete an object in an S3 bucket.
        /*!
          \param bucketName: The S3 bucket's name.
          \param key: The key for the object in the S3 bucket.
          \param client: An S3 client.
          \return bool: Function succeeded.
         */
        static bool
        deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
 &key,
                               Aws::S3::S3Client &client);
    }
}


//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
  \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
  \param saveFilePath: Path for saving a downloaded S3 object.
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &uploadFilePath,
                                           const Aws::String &saveFilePath,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: "doc-example-bucket-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = "doc-example-bucket-" +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(
```

```
Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                           clientConfig.region));
            request.WithCreateBucketConfiguration(createBucketConfiguration);
        }

        Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: createBucket: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
            return false;
        } else {
            std::cout << "Created the bucket, '" << bucketName <<
                      "', in the region, '" << clientConfig.region << "'." <<
std::endl;
        }
    }

    // 2. Upload a local file to the bucket.
    Aws::String key = "key-for-test";
    {
        Aws::S3::Model::PutObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        std::shared_ptr<Aws::FStream> input_data =
                Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                              uploadFilePath,
                                              std::ios_base::in |
                                              std::ios_base::binary);

        if (!input_data->is_open()) {
            std::cerr << "Error: unable to open file, '" << uploadFilePath <<
"'."
                      << std::endl;
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        }

        request.SetBody(input_data);
```

```
        Aws::S3::Model::PutObjectOutcome outcome =
                client.PutObject(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: putObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
            AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
            AwsDoc::S3::deleteBucket(bucketName, client);
            return false;
        } else {
            std::cout << "Added the object with the key, '" << key
                      << "', to the bucket, '"
                      << bucketName << "'." << std::endl;
        }
    }

    // 3. Download the object to a local file.
    {
        Aws::S3::Model::GetObjectRequest request;
        request.SetBucket(bucketName);
        request.SetKey(key);

        Aws::S3::Model::GetObjectOutcome outcome =
                client.GetObject(request);

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: getObject: " <<
                      err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        } else {
            std::cout << "Downloaded the object with the key, '" << key
                      << "', in the bucket, '"
                      << bucketName << "'." << std::endl;

            Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
                    GetBody();
            Aws::OFStream outStream(saveFilePath,
                                    std::ios_base::out | std::ios_base::binary);
            if (!outStream.is_open()) {
                std::cout << "Error: unable to open file, '" << saveFilePath <<
"'."
                          << std::endl;
```

```
            } else {
                outStream << ioStream.rdbuf();
                std::cout << "Wrote the downloaded object to the file '"
                          << saveFilePath << "'." << std::endl;
            }
        }
    }

    // 4. Copy the object to a different "folder" in the bucket.
    Aws::String copiedToKey = "test-folder/" + key;
    {
        Aws::S3::Model::CopyObjectRequest request;
        request.WithBucket(bucketName)
                .WithKey(copiedToKey)
                .WithCopySource(bucketName + "/" + key);

        Aws::S3::Model::CopyObjectOutcome outcome =
                client.CopyObject(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: copyObject: " <<
                      outcome.GetError().GetMessage() << std::endl;
        } else {
            std::cout << "Copied the object with the key, '" << key
                      << "', to the key, '" << copiedToKey
                      << ", in the bucket, '" << bucketName << "'." << std::endl;
        }
    }

    // 5. List objects in the bucket.
    {
        Aws::S3::Model::ListObjectsV2Request request;
        request.WithBucket(bucketName);

        Aws::String continuationToken;
        Aws::Vector<Aws::S3::Model::Object> allObjects;

        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
                    request);

            if (!outcome.IsSuccess()) {
```

```
                std::cerr << "Error: ListObjects: " <<
                          outcome.GetError().GetMessage() << std::endl;
                break;
            } else {
                Aws::Vector<Aws::S3::Model::Object> objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
 objects.end());
                continuationToken = outcome.GetResult().GetContinuationToken();
            }
        } while (!continuationToken.empty());

        std::cout << allObjects.size() << " objects in the bucket, '" <<
 bucketName
                  << "':" << std::endl;

        for (Aws::S3::Model::Object &object: allObjects) {
            std::cout << "     '" << object.GetKey() << "'" << std::endl;
        }
    }

    // 6. Delete all objects in the bucket.
    // All objects in the bucket must be deleted before deleting the bucket.
    AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
    AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

    // 7. Delete the bucket.
    return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
```

```
        std::cout << "Deleted the object with the key, '" << key
                  << "', from the bucket, '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
 &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅 AWS SDK for C++ API 参考中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

定义一个结构来包装此场景使用的桶和对象操作。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets() ([]types.Bucket, error) {
 result, err := basics.S3Client.ListBuckets(context.TODO(),
 &s3.ListBucketsInput{})
 var buckets []types.Bucket
 if err != nil {
  log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
 } else {
  buckets = result.Buckets
 }
 return buckets, err
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(bucketName string) (bool, error) {
 _, err := basics.S3Client.HeadBucket(context.TODO(), &s3.HeadBucketInput{
```

```
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(name string, region string) error {
 _, err := basics.S3Client.CreateBucket(context.TODO(), &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  log.Printf("Couldn't create bucket %v in Region %v. Here's why: %v\n",
   name, region, err)
 }
 return err
}



// UploadFile reads from a file and puts the data into an object in a bucket.
```

```
func (basics BucketBasics) UploadFile(bucketName string, objectKey string,
 fileName string) error {
 file, err := os.Open(fileName)
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(context.TODO(), &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
    fileName, bucketName, objectKey, err)
  }
 }
 return err
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }

 return err
```

```
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(bucketName string, objectKey string,
 fileName string) error {
 result, err := basics.S3Client.GetObject(context.TODO(), &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  return err
 }
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}



// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey
 string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
```

```
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}



// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(bucketName string, objectKey string,
 folderName string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(bucketName),
  CopySource: aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
  Key:        aws.String(fmt.Sprintf("%v/%v", folderName, objectKey)),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v/%v. Here's why: %v\n",
   bucketName, objectKey, bucketName, folderName, objectKey, err)
 }
 return err
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(sourceBucket string, destinationBucket
 string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v. Here's why: %v\n",
   sourceBucket, objectKey, destinationBucket, objectKey, err)
```

```
 }
 return err
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(bucketName string) ([]types.Object, error)
 {
 result, err := basics.S3Client.ListObjectsV2(context.TODO(),
 &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 })
 var contents []types.Object
 if err != nil {
  log.Printf("Couldn't list objects in bucket %v. Here's why: %v\n", bucketName,
 err)
 } else {
  contents = result.Contents
 }
 return contents, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(bucketName string, objectKeys []string)
 error {
 var objectIds []types.ObjectIdentifier
 for _, key := range objectKeys {
  objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
 }
 output, err := basics.S3Client.DeleteObjects(context.TODO(),
 &s3.DeleteObjectsInput{
  Bucket: aws.String(bucketName),
  Delete: &types.Delete{Objects: objectIds},
 })
 if err != nil {
  log.Printf("Couldn't delete objects from bucket %v. Here's why: %v\n",
 bucketName, err)
 } else {
  log.Printf("Deleted %v objects.\n", len(output.Deleted))
 }
 return err
```

```
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(bucketName string) error {
 _, err := basics.S3Client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
 }
 return err
}
```

运行交互式场景，向您展示如何使用 S3 存储桶和对象。

```
// RunGetStartedScenario is an interactive example that shows you how to use
 Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
 objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 4. Download an object to a local file.
// 5. Download a large object by using a download manager.
// 6. Copy an object to a different folder in the bucket.
// 7. List objects in the bucket.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
```

```
func RunGetStartedScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Println("Something went wrong with the demo.\n", r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 getting started demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}

 count := 10
 log.Printf("Let's list up to %v buckets for your account:", count)
 buckets, err := bucketBasics.ListBuckets()
 if err != nil {
  panic(err)
 }
 if len(buckets) == 0 {
  log.Println("You don't have any buckets!")
 } else {
  if count > len(buckets) {
   count = len(buckets)
  }
  for _, bucket := range buckets[:count] {
   log.Printf("\t%v\n", *bucket.Name)
  }
 }

 bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
 bucket:",
  demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(bucketName)
 if err != nil {
  panic(err)
 }
 if !bucketExists {
  err = bucketBasics.CreateBucket(bucketName, sdkConfig.Region)
  if err != nil {
   panic(err)
  } else {
```

```
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(bucketName, smallKey, smallFile)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
rand.Seed(time.Now().Unix())
rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(bucketName, largeKey, largeBytes)
if err != nil {
 panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(bucketName, smallKey, downloadFileName)
if err != nil {
 panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
```

```
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(bucketName, largeKey)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(bucketName, smallKey, folderName)
if err != nil {
 panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(bucketName)
if err != nil {
 panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
 objKeys = append(objKeys, *object.Key)
 log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
 "contents? (y/n)", "y") {
 log.Println("Deleting objects.")
 err = bucketBasics.DeleteObjects(bucketName, objKeys)
 if err != nil {
  panic(err)
 }
 log.Println("Deleting bucket.")
 err = bucketBasics.DeleteBucket(bucketName)
 if err != nil {
  panic(err)
 }
 log.Printf("Deleting downloaded file %v.\n", downloadFileName)
```

```
  err = os.Remove(downloadFileName)
  if err != nil {
   panic(err)
  }
 } else {
  log.Println("Okay. Don't forget to delete objects from your bucket to avoid
 charges.")
 }
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Deletes the object from the Amazon S3 bucket.
 * 8. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <bucketName> <key> <objectPath> <savePath> <toBucket>

                Where:
                    bucketName - The Amazon S3 bucket to create.
                    key - The key to use.
                    objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).
                    savePath - The path where the file is saved after it's
 downloaded (for example, C:/AWS/book2.pdf).
                    toBucket - An Amazon S3 bucket to where an object is copied
 to (for example, C:/AWS/book2.pdf).\s
                    """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Amazon S3 bucket.");
        createBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update a local file to the Amazon S3 bucket.");
        uploadLocalFile(s3, bucketName, key, objectPath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Download the object to another local file.");
        getObjectBytes(s3, bucketName, key, savePath);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Perform a multipart upload.");
        String multipartKey = "multiPartKey";
        multipartUpload(s3, toBucket, multipartKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List all objects located in the Amazon S3
bucket.");
        listAllObjects(s3, bucketName);
        anotherListExample(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Copy the object to another Amazon S3 bucket.");
```

```
        copyBucketObject(s3, bucketName, key, toBucket);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the object from the Amazon S3 bucket.");
        deleteObjectFromBucket(s3, bucketName, key);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Delete the Amazon S3 bucket.");
        deleteBucket(s3, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon S3 operations were successfully
performed");
        System.out.println(DASHES);
        s3.close();
    }

    // Create a bucket by using a S3Waiter object.
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteBucket(S3Client client, String bucket) {
       DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
               .bucket(bucket)
               .build();

       client.deleteBucket(deleteBucketRequest);
       System.out.println(bucket + " was deleted.");
   }

   /**
    * Upload an object in parts.
    */
   public static void multipartUpload(S3Client s3, String bucketName, String
key) {
       int mB = 1024 * 1024;
       // First create a multipart upload and get the upload id.
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();

       CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
       String uploadId = response.uploadId();
       System.out.println(uploadId);

       // Upload all the different parts of the object.
       UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
               .bucket(bucketName)
               .key(key)
               .uploadId(uploadId)
               .partNumber(1).build();

       String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
               .eTag();
       CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

       UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
               .uploadId(uploadId)
```

```
                .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
                .eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all
uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .multipartUpload(completedMultipartUpload)
                .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        byte[] b = new byte[size];
        new Random().nextBytes(b);
        return ByteBuffer.wrap(b);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();
```

```
            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void uploadLocalFile(S3Client s3, String bucketName, String
key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
    }

    public static void listAllObjects(S3Client s3, String bucketName) {
        ListObjectsV2Request listObjectsReqManual =
ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }
```

```
            listObjectsReqManual = listObjectsReqManual.toBuilder()
                    .continuationToken(listObjResponse.nextContinuationToken())
                    .build();
        }
    }

    public static void anotherListExample(S3Client s3, String bucketName) {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

        // Process response pages.
        listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

        // Helper method to work with paginated collection of items directly.
        listRes.contents().stream()
                .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

        for (S3Object content : listRes.contents()) {
            System.out.println(" Key: " + content.key() + " size = " +
content.size());
        }
    }

    public static void deleteObjectFromBucket(S3Client s3, String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        s3.deleteObject(deleteObjectRequest);
        System.out.println(key + " was deleted");
    }
```

```
    public static String copyBucketObject(S3Client s3, String fromBucket, String
 objectKey, String toBucket) {
        String encodedUrl = null;
        try {
            encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
 StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException e) {
            System.out.println("URL could not be encoded: " + e.getMessage());
        }
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .copySource(encodedUrl)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            System.out.println("The " + objectKey + " was copied to " +
 toBucket);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

首先，导入所有必需的模块。

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "url";
import { readdirSync, readFileSync, writeFileSync } from "fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

前面的导入引用了一些帮助程序实用程序。这些实用程序是在本节开头链接的 GitHub 存储库的本地实用程序。为了便于参考，请参阅这些实用程序的以下实现。

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox } from "@inquirer/prompts";

export class Prompter {
```

```
  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && prompt + " ";
    let ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };

  /**
   * @param {{ message: string }} options
   */
  confirm(options) {
    return confirm(options);
  }

  /**
   * @param {{ message: string, choices: { name: string, value: string }[]}}
 options
   */
  checkbox(options) {
    return checkbox(options);
  }
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
```

```
  return `${rule}\n    ${text}\n${rule}\n`;
};
```

S3 中的对象存储在“存储桶”中。让我们定义一个用于创建新存储桶的函数。

```
export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};
```

存储桶包含“对象”。此函数将目录的内容作为对象上传到您的存储桶。

```
export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {
    const filePath = `${folderPath}/${key}`;
    const fileContent = readFileSync(filePath);
    return {
      Key: key,
      Body: fileContent,
    };
  });

  for (let file of files) {
    await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketName,
        Body: file.Body,
        Key: file.Key,
      }),
    );
    console.log(`${file.Key} uploaded successfully.`);
  }
};
```

上传对象后，检查以确认它们已正确上传。可以使用 ListObjects 完成此操作。您将使用“Key”属性，但响应中还有其他有用的属性。

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(contentsList + "\n");
};
```

有时，您可能想要将对象从一个桶复制到另一个桶。为此，请使用 CopyObject 命令。

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });

  if (!proceed) {
    return;
  } else {
    const copy = async () => {
      try {
        const sourceBucket = await prompter.input({
          message: "Enter source bucket name:",
        });
        const sourceKey = await prompter.input({
          message: "Enter source key:",
        });
        const destinationKey = await prompter.input({
          message: "Enter destination key:",
        });

        const command = new CopyObjectCommand({
          Bucket: destinationBucket,
          CopySource: `${sourceBucket}/${sourceKey}`,
          Key: destinationKey,
        });
        await s3Client.send(command);
        await copyFileFromBucket({ destinationBucket });
```

```
      } catch (err) {
        console.error(`Copy error.`);
        console.error(err);
        const retryAnswer = await prompter.confirm({ message: "Try again?" });
        if (retryAnswer) {
          await copy();
        }
      }
    };
    await copy();
  }
};
```

没有用于从桶中获取多个对象的 SDK 方法。相反，您将创建一个要下载的对象列表并对其进行迭代。

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
    message: "Enter destination path for files:",
  });

  for (let content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};
```

是时候清除资源了。桶必须为空才能删除它。这两个函数清空并删除桶。

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
```

```
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

“main”函数将所有内容整合在一起。如果您直接运行此文件，将调用 main 函数。

```
const main = async () => {
  const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
    import.meta.url,
  )}../../../../resources/sample_files/.sample_media`;

  try {
    console.log(wrapText("Welcome to the Amazon S3 getting started example."));
    console.log("Let's create a bucket.");
    const bucketName = await createBucket();
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("File upload."));
    console.log(
      "I have some default files ready to go. You can edit the source code to
 provide your own.",
    );
    await uploadFilesToBucket({
      bucketName,
      folderPath: OBJECT_DIRECTORY,
    });

    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });
```

```
    console.log(wrapText("Copy files."));
    await copyFileFromBucket({ destinationBucket: bucketName });
    await listFilesInBucket({ bucketName });
    await prompter.confirm({ message: continueMessage });

    console.log(wrapText("Download files."));
    await downloadFilesFromBucket({ bucketName });

    console.log(wrapText("Clean up."));
    await emptyBucket({ bucketName });
    await deleteBucket({ bucketName });
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun main(args: Array<String>) {
    val usage = """
```

```
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
 example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for
 example, C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
```

```
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
```

```
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
```

```
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
```

```
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        echo("\n");
        echo("---------------------------------------\n");
        print("Welcome to the Amazon S3 getting started demo using PHP!\n");
        echo("---------------------------------------\n");

        $region = 'us-west-2';

        $this->s3client = new S3Client([
                'region' => $region,
        ]);
        /* Inline declaration example
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
        */

        $this->bucketName = "doc-example-bucket-" . uniqid();
```

```
        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
```

```
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
 $exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
```

```
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
 " . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }

        echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- 有关 API 详细信息，请参阅 AWS SDK for PHP API 参考中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)

    bucket_name = f"doc-example-bucket-{uuid.uuid4()}"
    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                "LocationConstraint": s3_resource.meta.client.meta.region_name
            }
        )
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f"Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")
        print(f"\nCan't continue the demo without a bucket!")
        return

    file_name = None
    while file_name is None:
```

```
        file_name = input("\nEnter a file you want to upload to your bucket: ")
        if not os.path.exists(file_name):
            print(f"Couldn't find file {file_name}. Are you sure it exists?")
            file_name = None

    obj = bucket.Object(os.path.basename(file_name))
    try:
        obj.upload_file(file_name)
        print(
            f"Uploaded file {file_name} into bucket {bucket.name} with key
 {obj.key}."
        )
    except S3UploadFailedError as err:
        print(f"Couldn't upload file {file_name} to {bucket.name}.")
        print(f"\t{err}")

    answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
    if answer.lower() == "y":
        data = io.BytesIO()
        try:
            obj.download_fileobj(data)
            data.seek(0)
            print(f"Got your object. Here are the first 20 bytes:\n")
            print(f"\t{data.read(20)}")
        except ClientError as err:
            print(f"Couldn't download {obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
```

```
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
 "
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject

- ListObjectsV2
- PutObject

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "doc-example-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: "us-east-1" # Note: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Tried and failed to create demo bucket.")
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
```

```
  else
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open("demo.txt", w) { |f| f.write("This is a demo file.") }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
  def upload_file(bucket)
    File.open("demo.txt", "w+") { |f| f.write("This is a demo file.") }
    s3_object = bucket.object(File.basename("demo.txt"))
    s3_object.upload_file("demo.txt")
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't upload file demo.txt to #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    s3_object
  end

  # Downloads an Amazon S3 object to a file.
  #
  # @param s3_object [Aws::S3::Object] The object to download.
  def download_file(s3_object)
    puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      puts("Enter a name for the downloaded file: ")
      file_name = gets.chomp
      s3_object.download_file(file_name)
      puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
    end
  rescue Aws::Errors::ServiceError => e
```

```
    puts("Couldn't download #{s3_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your
 bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == "y"
      dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
```

```
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
 ")
    answer = gets.chomp.downcase
    if answer == "y"
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the Amazon S3 getting started demo!")
  puts("-" * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError
  puts("Something went wrong with the demo!")
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
 == __FILE__
```

- 有关 API 详细信息，请参阅 AWS SDK for Ruby API 参考中的以下主题。
  - CopyObject
  - CreateBucket

- DeleteBucket
- DeleteObjects
- GetObject
- ListObjectsV2
- PutObject

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

用于运行场景的二进制 crate 的代码。

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_service::error::Error;
use uuid::Uuid;


#[tokio::main]
async fn main() -> Result<(), Error> {
    let (region, client, bucket_name, file_name, key, target_key) =
 initialize_variables().await;

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
 key, target_key).await
    {
        println!("{:?}", e);
    };

    Ok(())
}
```

```
async fn initialize_variables() -> (Region, Client, String, String, String,
 String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());

    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    (region, client, bucket_name, file_name, key, target_key)
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), Error> {
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;
    s3_service::upload_object(&client, &bucket_name, &file_name, &key).await?;
    let _object = s3_service::download_object(&client, &bucket_name, &key).await;
    s3_service::copy_object(&client, &bucket_name, &key, &target_key).await?;
    s3_service::list_objects(&client, &bucket_name).await?;
    s3_service::delete_objects(&client, &bucket_name).await?;
    s3_service::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}
```

包含二进制文件调用的常见操作的库 crate。

```
use aws_sdk_s3::operation::{
    copy_object::{CopyObjectError, CopyObjectOutput},
    create_bucket::{CreateBucketError, CreateBucketOutput},
    get_object::{GetObjectError, GetObjectOutput},
    list_objects_v2::ListObjectsV2Output,
    put_object::{PutObjectError, PutObjectOutput},
};
use aws_sdk_s3::types::{
    BucketLocationConstraint, CreateBucketConfiguration, Delete,
 ObjectIdentifier,
};
use aws_sdk_s3::{error::SdkError, primitives::ByteStream, Client};
use error::Error;
use std::path::Path;
use std::str;

pub mod error;

pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(),
 Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}

pub async fn delete_objects(client: &Client, bucket_name: &str) ->
 Result<Vec<String>, Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build()
            .map_err(Error::from)?;
        delete_objects.push(obj_id);
    }

    let return_keys = delete_objects.iter().map(|o| o.key.clone()).collect();

    if !delete_objects.is_empty() {
        client
            .delete_objects()
            .bucket(bucket_name)
```

```
            .delete(
                Delete::builder()
                    .set_objects(Some(delete_objects))
                    .build()
                    .map_err(Error::from)?,
            )
            .send()
            .await?;
    }

    let objects: ListObjectsV2Output =
 client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(Error::unhandled(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn list_objects(client: &Client, bucket: &str) -> Result<(), Error> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }
```

```
    Ok(())
}

pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<CopyObjectOutput, SdkError<CopyObjectError>> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await
}

pub async fn download_object(
    client: &Client,
    bucket_name: &str,
    key: &str,
) -> Result<GetObjectOutput, SdkError<GetObjectError>> {
    client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await
}

pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<PutObjectOutput, SdkError<PutObjectError>> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
```

```
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
}

pub async fn create_bucket(
    client: &Client,
    bucket_name: &str,
    region: &str,
) -> Result<CreateBucketOutput, SdkError<CreateBucketError>> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
  CATCH /aws1/cx_s3_bucketalrdyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
  CATCH /aws1/cx_s3_bktalrdyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.


"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
    READ DATASET iv_key INTO lv_file_content.
    CLOSE DATASET iv_key.

    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_file_content
    ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
    ENDTRY.

    " Get an object from a bucket. "
    TRY.
        DATA(lo_result) = lo_s3->getobject(
                    iv_bucket = iv_bucket_name
                    iv_key = iv_key
                ).
        DATA(lv_object_data) = lo_result->get_body( ).
        MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " Copy an object to a subfolder in a bucket. "
    TRY.
        lo_s3->copyobject(
          iv_bucket = iv_bucket_name
          iv_key = |{ iv_copy_to_folder }/{ iv_key }|
          iv_copysource = |{ iv_bucket_name }/{ iv_key }|
        ).
        MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
      CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
    ENDTRY.

    " List objects in the bucket. "
    TRY.
        DATA(lo_list) = lo_s3->listobjects(
           iv_bucket = iv_bucket_name
         ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
    DATA text TYPE string VALUE 'Object List - '.
    DATA lv_object_key TYPE /aws1/s3_objectkey.
    LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
      lv_object_key = lo_object->get_key( ).
      CONCATENATE lv_object_key ', ' INTO text.
```

```
    ENDLOOP.
    MESSAGE text TYPE'I'.

    " Delete the objects in a bucket. "
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_key
        ).
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        ).
        MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.


    " Delete the bucket. "
    TRY.
        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅适用于 SAP ABAP 的 AWS SDK 的 API 参考中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

一个 Swift 类，用于处理对 SDK for Swift 的调用。

```
import Foundation
import AWSS3
import ClientRuntime
import AWSClientRuntime
import Smithy

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let client: S3Client

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
 drive the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    ///            execute AWS operations.
    public init() async {
        do {
            client = try S3Client(region: "us-east-2")
        } catch {
            print("ERROR: ", dump(error, name: "Initializing S3 client"))
            exit(1)
        }
    }
```

```
    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    /// Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        let config = S3ClientTypes.CreateBucketConfiguration(
            locationConstraint: .usEast2
        )
        let input = CreateBucketInput(
            bucket: name,
            createBucketConfiguration: config
        )
        _ = try await client.createBucket(input: input)
    }

    /// Delete a bucket.
    /// - Parameter name: Name of the bucket to delete.
    public func deleteBucket(name: String) async throws {
        let input = DeleteBucketInput(
            bucket: name
        )
        _ = try await client.deleteBucket(input: input)
    }

    /// Upload a file from local storage to the bucket.
    /// - Parameters:
    ///   - bucket: Name of the bucket to upload the file to.
    ///   - key: Name of the file to create.
    ///   - file: Path name of the file to upload.
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
        let fileUrl = URL(fileURLWithPath: file)
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }
```

```
    /// Create a file in the specified bucket with the given name. The new
    /// file's contents are uploaded from a `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket to create a file in.
    ///   - key: Name of the file to create.
    ///   - data: A `Data` object to write into the new file.
    public func createFile(bucket: String, key: String, withData data: Data)
async throws {
        let dataStream = ByteStream.data(data)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }

    /// Download the named file to the given directory on the local device.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket that contains the file to be copied.
    ///   - key: The name of the file to copy from the bucket.
    ///   - to: The path of the directory on the local device where you want to
    ///     download the file.
    public func downloadFile(bucket: String, key: String, to: String) async
throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the data stream object. Return immediately if there isn't one.
        guard let body = output.body,
              let data = try await body.readData() else {
            return
        }
        try data.write(to: fileUrl)
    }
```

```
    /// Read the specified file from the given S3 bucket into a Swift
    /// `Data` object.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to read.
    ///   - key: Name of the file within the bucket to read.
    ///
    /// - Returns: A `Data` object containing the complete file data.
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the stream and return its contents in a `Data` object. If
        // there is no stream, return an empty `Data` object instead.
        guard let body = output.body,
              let data = try await body.readData() else {
            return "".data(using: .utf8)!
        }

        return data
    }

    /// Copy a file from one bucket to another.
    ///
    /// - Parameters:
    ///   - sourceBucket: Name of the bucket containing the source file.
    ///   - name: Name of the source file.
    ///   - destBucket: Name of the bucket to copy the file into.
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        _ = try await client.copyObject(input: input)
    }
```

```
    /// Deletes the specified file from Amazon S3.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to delete.
    ///   - key: Name of the file to delete.
    ///
    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
        )

        do {
            _ = try await client.deleteObject(input: input)
        } catch {
            throw error
        }
    }

    /// Returns an array of strings, each naming one file in the
    /// specified bucket.
    ///
    /// - Parameter bucket: Name of the bucket to get a file listing for.
    /// - Returns: An array of `String` objects, each giving the name of
    ///            one file contained in the bucket.
    public func listBucketFiles(bucket: String) async throws -> [String] {
        let input = ListObjectsV2Input(
            bucket: bucket
        )
        let output = try await client.listObjectsV2(input: input)
        var names: [String] = []

        guard let objList = output.contents else {
            return []
        }

        for obj in objList {
            if let objName = obj.key {
                names.append(objName)
            }
        }

        return names
    }
```

```
}
```

用于管理 SDK 调用的 Swift 命令行程序。

```
import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
 bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: """
        Performs the following Amazon S3 commands:

        * `CreateBucket`
        * `PutObject`
        * `GetObject`
        * `CopyObject`
        * `ListObjects`
        * `DeleteObjects`
        * `DeleteBucket`
        """
    )
```

```
   /// Called by ``main()`` to do the actual running of the AWS
   /// example.
   func runAsync() async throws {
       let serviceHandler = await ServiceHandler()

       // 1. Create the bucket.
       print("Creating the bucket \(bucketName)...")
       try await serviceHandler.createBucket(name: bucketName)

       // 2. Upload a file to the bucket.
       print("Uploading the file \(uploadSource)...")
       try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

       // 3. Download the file.
       print("Downloading the file \(objName) to \(downloadDir)...")
       try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

       // 4. Copy the file to another bucket.
       print("Copying the file to the bucket \(destBucket)...")
       try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

       // 5. List the contents of the bucket.

       print("Getting a list of the files in the bucket \(bucketName)")
       let fileList = try await serviceHandler.listBucketFiles(bucket:
bucketName)
       let numFiles = fileList.count
       if numFiles != 0 {
           print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
\(bucketName):")
           for name in fileList {
               print("  \(name)")
           }
       } else {
           print("No files found in bucket \(bucketName)")
       }

       // 6. Delete the objects from the bucket.

       print("Deleting the file \(objName) from the bucket \(bucketName)...")
```

```
        try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
        print("Deleting the file \(objName) from the bucket \(destBucket)...")
        try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

        // 7. Delete the bucket.
        print("Deleting the bucket \(bucketName)...")
        try await serviceHandler.deleteBucket(name: bucketName)

        print("Done.")
    }
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 对 Amazon S3 执行的操作

以下代码示例演示了如何使用 AWS SDK 来执行各个 Amazon S3 操作。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Amazon S3 API，是必须在上下文中运行的较大型程序的代码节选。您可以在适用于使用 AWS 软件开发工具包的 Amazon S3 的场景 中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Simple Storage Service API 参考。

示例

- 将 AbortMultipartUpload 与 AWS SDK 或 CLI 配合使用
- 将 AbortMultipartUploads 与 AWS SDK 或 CLI 配合使用
- 将 CompleteMultipartUpload 与 AWS SDK 或 CLI 配合使用
- 将 CopyObject 与 AWS SDK 或 CLI 配合使用
- 将 CreateBucket 与 AWS SDK 或 CLI 配合使用
- 将 CreateMultiRegionAccessPoint 与 AWS SDK 或 CLI 配合使用
- 将 CreateMultipartUpload 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucket 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketAnalyticsConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketCors 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketEncryption 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketInventoryConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketLifecycle 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketMetricsConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketPolicy 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketReplication 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketTagging 与 AWS SDK 或 CLI 配合使用
- 将 DeleteBucketWebsite 与 AWS SDK 或 CLI 配合使用
- 将 DeleteObject 与 AWS SDK 或 CLI 配合使用

- 将 DeleteObjectTagging 与 AWS SDK 或 CLI 配合使用
- 将 DeleteObjects 与 AWS SDK 或 CLI 配合使用
- 将 DeletePublicAccessBlock 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketAccelerateConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketAcl 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketAnalyticsConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketCors 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketEncryption 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketInventoryConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketLocation 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketLogging 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketMetricsConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketNotification 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketPolicy 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketPolicyStatus 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketReplication 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketRequestPayment 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketTagging 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketVersioning 与 AWS SDK 或 CLI 配合使用
- 将 GetBucketWebsite 与 AWS SDK 或 CLI 配合使用
- 将 GetObject 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectAcl 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectAttributes 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectLegalHold 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectRetention 与 AWS SDK 或 CLI 配合使用
- 将 GetObjectTagging 与 AWS SDK 或 CLI 配合使用
- 将 GetPublicAccessBlock 与 AWS SDK 或 CLI 配合使用
- 将 HeadBucket 与 AWS SDK 或 CLI 配合使用

- 将 HeadObject 与 AWS SDK 或 CLI 配合使用
- 将 ListBucketAnalyticsConfigurations 与 AWS SDK 或 CLI 配合使用
- 将 ListBucketInventoryConfigurations 与 AWS SDK 或 CLI 配合使用
- 将 ListBuckets 与 AWS SDK 或 CLI 配合使用
- 将 ListMultipartUploads 与 AWS SDK 或 CLI 配合使用
- 将 ListObjectVersions 与 AWS SDK 或 CLI 配合使用
- 将 ListObjects 与 AWS SDK 或 CLI 配合使用
- 将 ListObjectsV2 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketAccelerateConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketAcl 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketCors 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketEncryption 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketLifecycleConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketLogging 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketNotification 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketNotificationConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketPolicy 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketReplication 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketRequestPayment 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketTagging 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketVersioning 与 AWS SDK 或 CLI 配合使用
- 将 PutBucketWebsite 与 AWS SDK 或 CLI 配合使用
- 将 PutObject 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectAcl 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectLegalHold 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectLockConfiguration 与 AWS SDK 或 CLI 配合使用
- 将 PutObjectRetention 与 AWS SDK 或 CLI 配合使用
- 将 RestoreObject 与 AWS SDK 或 CLI 配合使用
- 将 SelectObjectContent 与 AWS SDK 或 CLI 配合使用
- 将 UploadPart 与 AWS SDK 或 CLI 配合使用

将 **AbortMultipartUpload** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 AbortMultipartUpload。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 删除未完成的分段上传
- 使用 Amazon S3 对象完整性

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 AbortMultipartUpload。

CLI

AWS CLI

中止指定的分段上传

以下 abort-multipart-upload 命令中止存储桶 my-bucket 中键 multipart/01 的分段上传：

```
aws s3api abort-multipart-upload \
    --bucket my-bucket \
    --key multipart/01 \
    --upload-
id dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

此命令所需的上传 ID 由 create-multipart-upload 输出，也可以使用 list-multipart-uploads 进行检索。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 AbortMultipartUpload。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令中止在 5 天前创建的分段上传。

```
Remove-S3MultipartUpload -BucketName test-files -DaysBefore 5
```

示例 2：此命令中止在 2014 年 1 月 2 日之前创建的分段上传。

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "Thursday, January
 02, 2014"
```

示例 3：此命令中止在 2014 年 1 月 2 日 10:45:37 之前创建的分段上传。

```
Remove-S3MultipartUpload -BucketName test-files -InitiatedDate "2014/01/02
 10:45:37"
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 AbortMultipartUpload。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **AbortMultipartUploads** 与 AWS SDK 或 CLI 配合使用

以下代码示例显示如何使用 AbortMultipartUploads。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to use the Amazon Simple Storage Service
/// (Amazon S3) to stop a multi-part upload process using the Amazon S3
/// TransferUtility.
/// </summary>
```

```
    public class AbortMPU
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            await AbortMPUAsync(client, bucketName);
        }

        /// <summary>
        /// Cancels the multi-part copy process.
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// the TransferUtility object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the
        /// multi-part copy operation is in progress.</param>
        public static async Task AbortMPUAsync(IAmazonS3 client, string
bucketName)
        {
            try
            {
                var transferUtility = new TransferUtility(client);

                // Cancel all in-progress uploads initiated before the specified
date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 AbortMultipartUploads。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CompleteMultipartUpload** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CompleteMultipartUpload。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 执行分段复制
- 执行分段上传
- 使用校验和
- 使用 Amazon S3 对象完整性

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

  const Aws::String &key,

  const Aws::String &uploadID,
```

```
  const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 CompleteMultipartUpload。

CLI

AWS CLI

以下命令完成存储桶 my-bucket 中密钥 multipart/01 的分段上传：

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket my-bucket --key 'multipart/01' --upload-
id dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

此命令所需的上传 ID 由 create-multipart-upload 输出，也可以使用 list-multipart-uploads 进行检索。

上述命令中的分段上传选项采用 JSON 结构，用于描述分段上传中应重新组合成完整文件的各个部分。在此示例中，`file://` 前缀用于从名为 `mpustruct` 的本地文件夹中的文件加载 JSON 结构。

mpustruct：

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

每次使用 `upload-part` 命令上传分段时，都会输出每个分段的 ETag 值，也可以通过调用 `list-parts` 来检索，或者通过对每个分段进行 MD5 校验和执行计算。

输出：

```
{
    "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
    "Bucket": "my-bucket",
    "Location": "https://my-bucket.s3.amazonaws.com/multipart%2F01",
    "Key": "multipart/01"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 CompleteMultipartUpload。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await
    .unwrap();
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 CompleteMultipartUpload。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CopyObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CopyObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基础知识
- 加密入门

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

public class CopyObject
{
    public static async Task Main()
    {
        // Specify the AWS Region where your buckets are located if it is
        // different from the AWS Region of the default user.
        IAmazonS3 s3Client = new AmazonS3Client();

        // Remember to change these values to refer to your Amazon S3
objects.
        string sourceBucketName = "doc-example-bucket1";
        string destinationBucketName = "doc-example-bucket2";
        string sourceObjectKey = "testfile.txt";
        string destinationObjectKey = "testfilecopy.txt";

        Console.WriteLine($"Copying {sourceObjectKey} from {sourceBucketName}
to ");
        Console.WriteLine($"{destinationBucketName} as
{destinationObjectKey}");

        var response = await CopyingObjectAsync(
            s3Client,
            sourceObjectKey,
            destinationObjectKey,
            sourceBucketName,
            destinationBucketName);
```

```
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("\nCopy complete.");
            }
        }

        /// <summary>
        /// This method calls the AWS SDK for .NET to copy an
        /// object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The Amazon S3 client object.</param>
        /// <param name="sourceKey">The name of the object to be copied.</param>
        /// <param name="destinationKey">The name under which to save the copy.</
param>
        /// <param name="sourceBucketName">The name of the Amazon S3 bucket
        /// where the file is located now.</param>
        /// <param name="destinationBucketName">The name of the Amazon S3
        /// bucket where the copy should be saved.</param>
        /// <returns>Returns a CopyObjectResponse object with the results from
        /// the async call.</returns>
        public static async Task<CopyObjectResponse> CopyingObjectAsync(
            IAmazonS3 client,
            string sourceKey,
            string destinationKey,
            string sourceBucketName,
            string destinationBucketName)
        {
            var response = new CopyObjectResponse();
            try
            {
                var request = new CopyObjectRequest
                {
                    SourceBucket = sourceBucketName,
                    SourceKey = sourceKey,
                    DestinationBucket = destinationBucketName,
                    DestinationKey = destinationKey,
                };
                response = await client.CopyObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error copying object: '{ex.Message}'");
            }
```

```
            return response;
        }
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for .NET API 参考中的 CopyObject。

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file from and to.
#       $2 - The key of the source file to copy.
#       $3 - The key of the destination file.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_item_in_bucket() {
```

```
  local bucket_name=$1
  local source_key=$2
  local destination_key=$3
  local response

  response=$(aws s3api copy-object \
    --bucket "$bucket_name" \
    --copy-source "$bucket_name/$source_key" \
    --key "$destination_key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api copy-object operation failed.\n$response"
    return 1
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 CopyObject。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
 &fromBucket, const Aws::String &toBucket,
                             const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
            .WithKey(objectKey)
            .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
```

```
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;

    } else {
        std::cout << "Successfully copied " << objectKey << " from " <<
 fromBucket <<
                  " to " << toBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅 AWS SDK for C++ API 参考中的 CopyObject。

CLI

AWS CLI

以下命令将对象从 bucket-1 复制到 bucket-2：

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --
bucket bucket-2
```

输出：

```
{
    "CopyObjectResult": {
        "LastModified": "2015-11-10T01:07:25.000Z",
        "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
    },
    "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifMlkA"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 CopyObject。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(sourceBucket string, destinationBucket
 string, objectKey string) error {
 _, err := basics.S3Client.CopyObject(context.TODO(), &s3.CopyObjectInput{
  Bucket:     aws.String(destinationBucket),
  CopySource: aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
  Key:        aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't copy object from %v:%v to %v:%v. Here's why: %v\n",
   sourceBucket, objectKey, destinationBucket, objectKey, err)
 }
 return err
}
```

- 有关 API 详细信息，请参阅 AWS SDK for Go API 参考中的 CopyObject。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用 S3Client 复制对象。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <objectKey> <fromBucket> <toBucket>

                Where:
                    objectKey - The name of the object (for example, book.pdf).
                    fromBucket - The S3 bucket name that contains the object (for
 example, bucket1).
                    toBucket - The S3 bucket to copy the object to (for example,
 bucket2).
                """;
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String objectKey = args[0];
        String fromBucket = args[1];
        String toBucket = args[2];
        System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
 fromBucket, toBucket);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        copyBucketObject(s3, fromBucket, objectKey, toBucket);
        s3.close();
    }

    public static String copyBucketObject(S3Client s3, String fromBucket, String
 objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
                .sourceBucket(fromBucket)
                .sourceKey(objectKey)
                .destinationBucket(toBucket)
                .destinationKey(objectKey)
                .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            return copyRes.copyObjectResult().toString();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

使用 S3TransferManager 从一个桶将对象复制到另一个桶。查看完整文件并进行测试。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String
 bucketName,
            String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
                .sourceBucket(bucketName)
                .sourceKey(key)
                .destinationBucket(destinationBucket)
                .destinationKey(destinationKey)
                .build();

        CopyRequest copyRequest = CopyRequest.builder()
                .copyObjectRequest(copyObjectRequest)
                .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 CopyObject。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

复制对象。

```
import { S3Client, CopyObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new CopyObjectCommand({
    CopySource: "SOURCE_BUCKET/SOURCE_OBJECT_KEY",
    Bucket: "DESTINATION_BUCKET",
    Key: "NEW_OBJECT_KEY",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 详细信息，请参阅 AWS SDK for JavaScript API 参考中的 CopyObject。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
 StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 CopyObject。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

对象的简单副本。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $folder = "copied-folder";
            $this->s3client->copyObject([
                'Bucket' => $this->bucketName,
                'CopySource' => "$this->bucketName/$fileName",
                'Key' => "$folder/$fileName-copy",
            ]);
            echo "Copied $fileName to $folder/$fileName-copy.\n";
        } catch (Exception $exception) {
            echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with object copying before continuing.");
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for PHP API 参考中的 CopyObject。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令将对象“sample.txt”从存储桶“test-files”复制到同一个存储桶，但新键为“sample-copy.txt”。

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-
copy.txt
```

示例 2：此命令将对象“sample.txt”从存储桶“test-files”复制到存储桶“backup-files”，键为“sample-copy.txt”。

```
Copy-S3Object -BucketName test-files -Key sample.txt -DestinationKey sample-
copy.txt -DestinationBucket backup-files
```

示例 3：此命令将对象“sample.txt”从存储桶“test-files”下载到名为“local-sample.txt”的本地文件。

```
Copy-S3Object -BucketName test-files -Key sample.txt -LocalFile local-sample.txt
```

示例 4：将单个对象下载到指定的文件。下载的文件可以在 c:\downloads\data\archive.zip 中找到

```
Copy-S3Object -BucketName test-files -Key data/archive.zip -LocalFolder c:
\downloads
```

示例 5：将与指定的键前缀匹配的所有对象下载到本地文件夹。相对键层次结构将作为子文件夹保留在总体下载位置中。

```
Copy-S3Object -BucketName test-files -KeyPrefix data -LocalFolder c:\downloads
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 CopyObject。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
```

```
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 CopyObject。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

复制对象。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

复制对象并向目标对象添加服务器端加密。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
 target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
```

```
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
 why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
 target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅 AWS SDK for Ruby API 参考中的 CopyObject。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<CopyObjectOutput, SdkError<CopyObjectError>> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 CopyObject。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
TRY.
    lo_s3->copyobject(
      iv_bucket = iv_dest_bucket
      iv_key = iv_dest_object
      iv_copysource = |{ iv_src_bucket }/{ iv_src_object }|
    ).
    MESSAGE 'Object copied to another bucket.' TYPE 'I'.
  CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
  CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API 参考》中的 CopyObject。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
 String) async throws {
        let srcUrl = ("\(sourceBucket)/
\(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        _ = try await client.copyObject(input: input)
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API》中的 CopyObject。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateBucket** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateBucket`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基础知识
- 处理版本控制对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Shows how to create a new Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value representing the success or failure of
        /// the bucket creation process.</returns>
        public static async Task<bool> CreateBucketAsync(IAmazonS3 client, string
bucketName)
        {
            try
            {
                var request = new PutBucketRequest
                {
                    BucketName = bucketName,
                    UseClientRegion = true,
                };

                var response = await client.PutBucketAsync(request);
                return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error creating bucket: '{ex.Message}'");
                return false;
            }
        }
```

创建一个启用对象锁定的存储桶。

```
    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
    {
```

```
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for .NET API 参考中的 CreateBucket。

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
```

```
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

########################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
########################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

########################################################################
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#       -b bucket_name  -- The name of the bucket to create.
#       -r region_code  -- The code for an AWS Region in which to
#                          create the bucket.
#
# Returns:
#       The URL of the bucket that was created.
#     And:
#       0 - If successful.
#       1 - If it fails.
########################################################################
function create_bucket() {
  local bucket_name region_code response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function create_bucket"
    echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
    echo "  -b bucket_name    The name of the bucket. It must be globally
 unique."
    echo "  [-r region_code]    The code for an AWS Region in which the bucket is
 created."
    echo ""
```

```
  }

  # Retrieve the calling parameters.
  while getopts "b:r:h" option; do
    case "${option}" in
      b) bucket_name="${OPTARG}" ;;
      r) region_code="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
  fi

  local bucket_config_arg
  # A location constraint for "us-east-1" returns an error.
  if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
  fi

  iecho "Parameters:\n"
  iecho "    Bucket name:   $bucket_name"
  iecho "    Region code:   $region_code"
  iecho ""

  # If the bucket already exists, we don't want to try to create it.
  if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
  fi

  # shellcheck disable=SC2086
```

```
  response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 CreateBucket。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(

 Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                         clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
```

```
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
                  " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅 AWS SDK for C++ API 参考中的 CreateBucket。

CLI

AWS CLI

示例 1：创建存储桶

以下 create-bucket 示例创建一个名为 my-bucket 的存储桶：

```
aws s3api create-bucket \
    --bucket my-bucket \
    --region us-east-1
```

输出：

```
{
    "Location": "/my-bucket"
}
```

有关更多信息，请参阅《Amazon S3 用户指南》中的创建存储桶。

示例 2：创建带有强制拥有者的存储桶

以下 create-bucket 示例创建一个名为 my-bucket 的存储桶，该存储桶对于 S3 对象所有权使用强制存储桶拥有者设置。

```
aws s3api create-bucket \
```

```
    --bucket my-bucket \
    --region us-east-1 \
    --object-ownership BucketOwnerEnforced
```

输出：

```
{
    "Location": "/my-bucket"
}
```

有关更多信息，请参阅《Amazon S3 用户指南》中的控制对象所有权和禁用 ACL。

示例 3：在“us-east-1”区域之外创建存储桶

以下 `create-bucket` 示例在 `eu-west-1` 区域中创建名为 `my-bucket` 的存储桶。`us-east-1` 之外的区域需要指定相应的 `LocationConstraint` 才能在所需区域创建存储桶。

```
aws s3api create-bucket \
    --bucket my-bucket \
    --region eu-west-1 \
    --create-bucket-configuration LocationConstraint=eu-west-1
```

输出：

```
{
    "Location": "http://my-bucket.s3.amazonaws.com/"
}
```

有关更多信息，请参阅《Amazon S3 用户指南》中的创建存储桶。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 CreateBucket。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用默认配置创建存储桶。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(name string, region string) error {
 _, err := basics.S3Client.CreateBucket(context.TODO(), &s3.CreateBucketInput{
  Bucket: aws.String(name),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 })
 if err != nil {
  log.Printf("Couldn't create bucket %v in Region %v. Here's why: %v\n",
   name, region, err)
 }
 return err
}
```

创建带有对象锁定功能的存储桶，并等待该存储桶创建成功。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}
```

```
// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 CreateBucket。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建存储桶。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the bucket to create. The bucket
 name must be unique, or an error occurs.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Creating a bucket named %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        createBucket(s3, bucketName);
        s3.close();
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
 s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

创建一个启用对象锁定的存储桶。

```
    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
 bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 CreateBucket。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建存储桶。

```
import { CreateBucketCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new CreateBucketCommand({
```

```
    // The name of the bucket. Bucket names are unique and have several other
 constraints.
    // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/
bucketnamingrules.html
    Bucket: "bucket-name",
  });

  try {
    const { Location } = await client.send(command);
    console.log(`Bucket created with location ${Location}`);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 CreateBucket。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun createNewBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 CreateBucket。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建存储桶。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->createBucket([
                'Bucket' => $this->bucketName,
                'CreateBucketConfiguration' => ['LocationConstraint' => $region],
            ]);
            echo "Created bucket named: $this->bucketName \n";
        } catch (Exception $exception) {
            echo "Failed to create bucket $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with bucket creation before continuing.");
        }
```

- 有关 API 详细信息，请参阅 AWS SDK for PHP API 参考中的 CreateBucket。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用默认设置创建存储桶。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in
 the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
 is
                                not specified, the Region configured in your
 shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
 region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
 region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
```

```
            raise error
```

使用生命周期配置创建版本控制的桶。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
```

```
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 CreateBucket。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                  create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
```

```
      @bucket.client.get_bucket_location(bucket:
 @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅 AWS SDK for Ruby API 参考中的 CreateBucket。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
pub async fn create_bucket(
    client: &Client,
    bucket_name: &str,
    region: &str,
) -> Result<CreateBucketOutput, SdkError<CreateBucketError>> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
```

```
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 CreateBucket。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        lo_s3->createbucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'S3 bucket created.' TYPE 'I'.
      CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
      CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API 参考》中的 CreateBucket。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
public func createBucket(name: String) async throws {
    let config = S3ClientTypes.CreateBucketConfiguration(
        locationConstraint: .usEast2
    )
    let input = CreateBucketInput(
        bucket: name,
        createBucketConfiguration: config
    )
    _ = try await client.createBucket(input: input)
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 CreateBucket。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateMultiRegionAccessPoint** 与 AWS SDK 或 CLI 配合使用

以下代码示例显示如何使用 `CreateMultiRegionAccessPoint`。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

将 S3 控制客户端配置为向 us-west-2 区域发送请求。

```
        suspend fun createS3ControlClient(): S3ControlClient {
            // Configure your S3ControlClient to send requests to US West
(Oregon).
            val s3Control = S3ControlClient.fromEnvironment {
                region = "us-west-2"
            }
            return s3Control
        }
```

创建多区域接入点。

```
    suspend fun createMrap(
        s3Control: S3ControlClient,
        accountIdParam: String,
        bucketName1: String,
        bucketName2: String,
        mrapName: String,
    ): String {
        println("Creating MRAP ...")
        val createMrapResponse: CreateMultiRegionAccessPointResponse =
            s3Control.createMultiRegionAccessPoint {
                accountId = accountIdParam
                clientToken = UUID.randomUUID().toString()
                details {
                    name = mrapName
                    regions = listOf(
                        Region {
                            bucket = bucketName1
                        },
```

```
                        Region {
                            bucket = bucketName2
                        },
                    )
                }
            }
        val requestToken: String? = createMrapResponse.requestTokenArn

        // Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
        if (requestToken != null) {
            waitForSucceededStatus(s3Control, requestToken, accountIdParam)
            println("MRAP created")
        }

        val getMrapResponse =
            s3Control.getMultiRegionAccessPoint(
                input = GetMultiRegionAccessPointRequest {
                    accountId = accountIdParam
                    name = mrapName
                },
            )
        val mrapAlias = getMrapResponse.accessPoint?.alias
        return "arn:aws:s3::$accountIdParam:accesspoint/$mrapAlias"
    }
```

等待多区域接入点变为可用状态。

```
        suspend fun waitForSucceededStatus(
            s3Control: S3ControlClient,
            requestToken: String,
            accountIdParam: String,
            timeBetweenChecks: Duration = 1.minutes,
        ) {
            var describeResponse: DescribeMultiRegionAccessPointOperationResponse
            describeResponse = s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )
```

```
            var status: String? = describeResponse.asyncOperation?.requestStatus
            while (status != "SUCCEEDED") {
                delay(timeBetweenChecks)
                describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
                    input = DescribeMultiRegionAccessPointOperationRequest {
                        accountId = accountIdParam
                        requestTokenArn = requestToken
                    },
                )
                status = describeResponse.asyncOperation?.requestStatus
                println(status)
            }
        }
```

- 有关更多信息，请参阅 AWS SDK for Kotlin 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 CreateMultiRegionAccessPoint。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateMultipartUpload** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateMultipartUpload`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 执行分段复制
- 执行分段上传
- 使用校验和
- 使用 Amazon S3 对象完整性

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
 &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return uploadID;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 CreateMultipartUpload。

CLI

AWS CLI

以下命令在存储桶 my-bucket 中创建带有密钥 multipart/01 的分段上传：

```
aws s3api create-multipart-upload --bucket my-bucket --key 'multipart/01'
```

输出：

```
{
    "Bucket": "my-bucket",
    "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
    "Key": "multipart/01"
}
```

完成的文件在存储桶 my-bucket 中名为 multipart 文件夹中将命名为 01。保存上传 ID、密钥和存储桶名称以供 upload-part 命令使用。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 CreateMultipartUpload。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
```

```
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await
        .unwrap();
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 CreateMultipartUpload。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucket** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucket。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基础知识

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Shows how to delete an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to
delete.</param>
        /// <returns>A boolean value that represents the success or failure of
        /// the delete operation.</returns>
```

```
        public static async Task<bool> DeleteBucketAsync(IAmazonS3 client, string
 bucketName)
        {
            var request = new DeleteBucketRequest
            {
                BucketName = bucketName,
            };

            var response = await client.DeleteBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
```

- 有关 API 详细信息，请参阅 AWS SDK for .NET API 参考中的 DeleteBucket。

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api delete-bucket \
    --bucket "$bucket_name")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
    return 1
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucket。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
```

```
            client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅 AWS SDK for C++ API 参考中的 DeleteBucket。

CLI

AWS CLI

以下命令删除名为 my-bucket 的存储桶：

```
aws s3api delete-bucket --bucket my-bucket --region us-east-1
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucket。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
```

```
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DeleteBucket deletes a bucket. The bucket must be empty or an error is
 returned.
func (basics BucketBasics) DeleteBucket(bucketName string) error {
 _, err := basics.S3Client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{
  Bucket: aws.String(bucketName)})
 if err != nil {
  log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
 }
 return err
}
```

- 有关 API 详细信息，请参阅 AWS SDK for Go API 参考中的 DeleteBucket。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucket)
                .build();

        s3.deleteBucket(deleteBucketRequest);
        s3.close();
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 DeleteBucket。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

删除存储桶。

```
import { DeleteBucketCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Delete a bucket.
export const main = async () => {
  const command = new DeleteBucketCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 DeleteBucket。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

删除空桶。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $this->s3client->deleteBucket([
                'Bucket' => $this->bucketName,
            ]);
            echo "Deleted bucket $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
            exit("Please fix error with bucket deletion before continuing.");
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for PHP API 参考中的 DeleteBucket。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令从存储桶“test-files”中移除所有对象和对象版本，然后删除该存储桶。在继续操作之前，该命令将提示进行确认。添加 -Force 开关可禁止确认。请注意，不能删除不为空的存储桶。

```
Remove-S3Bucket -BucketName test-files -DeleteBucketContent
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteBucket。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 DeleteBucket。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
  answer = gets.chomp.downcase
  if answer == "y"
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- 有关 API 详细信息，请参阅 AWS SDK for Ruby API 参考中的 DeleteBucket。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(),
 Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 DeleteBucket。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.

        lo_s3->deletebucket(
            iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API 参考》中的 DeleteBucket。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    _ = try await client.deleteBucket(input: input)
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 DeleteBucket。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketAnalyticsConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketAnalyticsConfiguration。

CLI

AWS CLI

删除存储桶的分析配置

以下 delete-bucket-analytics-configuration 示例移除指定存储桶和 ID 的分析配置。

```
aws s3api delete-bucket-analytics-configuration \
    --bucket my-bucket \
    --id 1
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketAnalyticsConfiguration。

PowerShell

适用于 PowerShell 的工具

示例 1：该命令移除给定 S3 存储桶中名为“testfilter”的分析筛选条件。

```
Remove-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteBucketAnalyticsConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketCors** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketCors。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
```

```
        /// Deletes a CORS configuration from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to delete the CORS configuration from the bucket.</param>
        private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
        {
            DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            await client.DeleteCORSConfigurationAsync(request);
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 DeleteBucketCors。

CLI

AWS CLI

以下命令从名为 my-bucket 的存储桶中删除跨源资源共享配置：

```
aws s3api delete-bucket-cors --bucket my-bucket
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketCors。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
```

```
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
self.bucket.name)
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 DeleteBucketCors。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"
```

```
# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 DeleteBucketCors。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketEncryption** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketEncryption。

CLI

AWS CLI

删除存储桶的服务器端加密配置

以下 delete-bucket-encryption 示例删除指定存储桶的服务器端加密配置。

```
aws s3api delete-bucket-encryption \
    --bucket my-bucket
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketEncryption。

PowerShell

适用于 PowerShell 的工具

示例 1：这将禁用为所提供的 S3 存储桶启用的加密。

```
Remove-S3BucketEncryption -BucketName 's3casetestbucket'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
 target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteBucketEncryption。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketInventoryConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketInventoryConfiguration。

CLI

AWS CLI

删除存储桶的清单配置

以下 delete-bucket-inventory-configuration 示例删除指定存储桶的 ID 为 1 的清单配置。

```
aws s3api delete-bucket-inventory-configuration \
    --bucket my-bucket \
    --id 1
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketInventoryConfiguration。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令移除与给定 S3 存储桶相对应的名为“testInventoryName”的清单。

```
Remove-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testInventoryName'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
 (DeleteBucketInventoryConfiguration)" on target "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteBucketInventoryConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketLifecycle** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketLifecycle。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// This method removes the Lifecycle configuration from the named
        /// S3 bucket.
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the RemoveLifecycleConfigAsync method.</param>
        /// <param name="bucketName">A string representing the name of the
        /// S3 bucket from which the configuration will be removed.</param>
        public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
string bucketName)
        {
            var request = new DeleteLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 DeleteBucketLifecycle。

CLI

AWS CLI

以下命令从名为 my-bucket 的存储桶中删除生命周期配置：

```
aws s3api delete-bucket-lifecycle --bucket my-bucket
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketLifecycle。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.",
 self.bucket.name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
            )
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 DeleteBucketLifecycle。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketMetricsConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketMetricsConfiguration。

CLI

AWS CLI

删除存储桶的指标配置

以下 delete-bucket-metrics-configuration 示例移除指定存储桶和 ID 的指标配置。

```
aws s3api delete-bucket-metrics-configuration \
    --bucket my-bucket \
    --id 123
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketMetricsConfiguration。

PowerShell

适用于 PowerShell 的工具

示例 1：该命令移除给定 S3 存储桶中名为“testmetrics”的指标筛选条件。

```
Remove-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testmetrics'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteBucketMetricsConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketPolicy** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketPolicy。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                    const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
 client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 DeleteBucketPolicy。

CLI

AWS CLI

以下命令从名为 my-bucket 的存储桶中删除存储桶策略：

```
aws s3api delete-bucket-policy --bucket my-bucket
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketPolicy。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """

                Usage:
```

```
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from
 (for example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    // Delete the bucket policy.
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 DeleteBucketPolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

删除存储桶策略。

```
import { DeleteBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// This will remove the policy from the bucket.
export const main = async () => {
  const command = new DeleteBucketPolicyCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 DeleteBucketPolicy。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 DeleteBucketPolicy。

PowerShell

适用于 PowerShell 的工具

示例 1：该命令移除与给定 S3 存储桶关联的存储桶策略。

```
Remove-S3BucketPolicy -BucketName 's3testbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteBucketPolicy。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 DeleteBucketPolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's
 why: #{e.message}"
    false
  end

end
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 DeleteBucketPolicy。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketReplication** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketReplication。

CLI

AWS CLI

以下命令从名为 my-bucket 的存储桶中删除复制配置：

```
aws s3api delete-bucket-replication --bucket my-bucket
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketReplication。

PowerShell

适用于 PowerShell 的工具

示例 1：删除与名为“mybucket”的存储桶关联的复制配置。请注意，此操作需要 s3:DeleteReplicationConfiguration 操作的权限。在操作继续之前，系统将提示您进行确认 - 要取消确认，请使用 -Force 开关。

```
Remove-S3BucketReplication -BucketName mybucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteBucketReplication。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketTagging** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketTagging。

CLI

AWS CLI

以下命令从名为 my-bucket 的存储桶中删除标记配置：

```
aws s3api delete-bucket-tagging --bucket my-bucket
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketTagging。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令移除与给定 S3 存储桶关联的所有标签。

```
Remove-S3BucketTagging -BucketName 's3testbucket'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteBucketTagging。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteBucketWebsite** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteBucketWebsite。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
 &clientConfig) {
```

```
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
            client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 DeleteBucketWebsite。

CLI

AWS CLI

以下命令从名为 my-bucket 的存储桶中删除网站配置：

```
aws s3api delete-bucket-website --bucket my-bucket
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteBucketWebsite。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the website
 configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket:
 %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }
```

```
    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
 {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 DeleteBucketWebsite。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

从存储桶中删除网站配置。

```
import { DeleteBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Disable static website hosting on the bucket.
export const main = async () => {
  const command = new DeleteBucketWebsiteCommand({
```

```
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 DeleteBucketWebsite。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令禁用给定 S3 存储桶的静态网站托管属性。

```
Remove-S3BucketWebsite -BucketName 's3testbucket'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
 "s3testbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteBucketWebsite。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 使用 Amazon S3 对象完整性
- 处理版本控制对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

删除不受版本控制的 S3 存储桶中的对象。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
    /// <summary>
    /// The Main method initializes the necessary variables and then calls
    /// the DeleteObjectNonVersionedBucketAsync method to delete the object
    /// named by the keyName parameter.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "doc-example-bucket";
```

```
            const string keyName = "testfile.txt";

            // If the Amazon S3 bucket is located in an AWS Region other than the
            // Region of the default account, define the AWS Region for the
            // Amazon S3 bucket in your call to the AmazonS3Client constructor.
            // For example RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();
            await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
        }

        /// <summary>
        /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
        /// desired object from the named bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to delete
        /// an object from an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// object will be deleted.</param>
        /// <param name="keyName">The name of the object to delete.</param>
        public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                Console.WriteLine($"Deleting object: {keyName}");
                await client.DeleteObjectAsync(deleteObjectRequest);
                Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{ex.Message}' when deleting an object.");
            }
        }
    }
```

删除受版本控制的 S3 存储桶中的对象。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example creates an object in an Amazon Simple Storage Service
    /// (Amazon S3) bucket and then deletes the object version that was
    /// created.
    /// </summary>
    public class DeleteObjectVersion
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "verstioned-object.txt";

            // If the AWS Region of the default user is different from the AWS
            // Region of the Amazon S3 bucket, pass the AWS Region of the
            // bucket region to the Amazon S3 client object's constructor.
            // Define it like this:
            //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
            IAmazonS3 client = new AmazonS3Client();

            await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// This method creates and then deletes a versioned object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// create and delete the object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// object will be created and deleted.</param>
        /// <param name="keyName">The key name of the object to create.</param>
        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
```

```
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName,
 keyName);

                // Delete the object by specifying an object key and a version
 ID.
                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// This method is used to create the temporary Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object which will be
 used
        /// to create the temporary Amazon S3 object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// will be created.</param>
        /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
        /// <returns>The Version ID of the created object.</returns>
        public static async Task<string> PutAnObject(IAmazonS3 client, string
 bucketName, string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!",
            };
```

```
            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
```

- 有关更多信息，请参阅《AWS SDK for .NET API 参考》中的 DeleteObject。

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - The key (file name) in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
```

```
function delete_item_in_bucket() {
  local bucket_name=$1
  local key=$2
  local response

  response=$(aws s3api delete-object \
    --bucket "$bucket_name" \
    --key "$key")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteObject。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                              const Aws::String &fromBucket,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
            .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
            client.DeleteObject(request);
```

```
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关更多信息，请参阅《AWS SDK for C++ API 参考》中的 DeleteObject。

CLI

AWS CLI

以下命令从名为 my-bucket 的存储桶中删除名为 test.txt 的对象：

```
aws s3api delete-object --bucket my-bucket --key test.txt
```

如果启用了存储桶版本控制，则输出将包含删除标记的版本 ID：

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tNDlWlGq",
  "DeleteMarker": true
}
```

有关删除对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“删除对象”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteObject。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
```

```
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
    }
   }
  }
 } else {
  deleted = true
 }
 return deleted, err
}
```

- 有关更多信息，请参阅《AWS SDK for Go API 参考》中的 DeleteObject。

JavaScript

适用于 JavaScript 的 SDK（v3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

删除对象。

```
import { DeleteObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new DeleteObjectCommand({
    Bucket: "test-bucket",
    Key: "test-key.txt",
  });
```

```
  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅《AWS SDK for JavaScript API 参考》中的 DeleteObject。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

删除对象。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def delete(self):
        """
        Deletes the object.
        """
        try:
```

```
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
```

通过删除对象的较高版本，将对象回滚到以前的版本。

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
```

```
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
 f"{object_key}."
        )
```

通过移除对象的活动删除标记来恢复已删除对象。

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
```

```
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

创建一个 Lambda 处理程序，从 S3 对象中移除删除标记。此处理程序可用于有效地清理版本控制的桶中无关的删除标记。

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of
 the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
 obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
 delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
```

```
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
 marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
 obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
 retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
```

```
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 DeleteObject。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn remove_object(client: &Client, bucket: &str, key: &str) -> Result<(),
 Error> {
    client
        .delete_object()
        .bucket(bucket)
```

```
        .key(key)
        .send()
        .await?;

    println!("Object deleted.");

    Ok(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 DeleteObject。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        lo_s3->deleteobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_object_key
        ).
        MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API 参考》中的 DeleteObject。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    } catch {
        throw error
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 DeleteObject。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteObjectTagging** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteObjectTagging。

CLI

AWS CLI

删除对象的标签集

以下 delete-object-tagging 示例从对象 doc1.rtf 中删除带有指定键的标签。

```
aws s3api delete-object-tagging \
    --bucket my-bucket \
    --key doc1.rtf
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteObjectTagging。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令移除给定 S3 存储桶中与键为“testfile.txt”的对象关联的所有标签。

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 's3testbucket' -Select
 '^Key'
```

输出：

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target
 "testfile.txt".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"): Y
testfile.txt
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteObjectTagging。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteObjects** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteObjects。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基础知识

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

从 S3 存储桶中删除所有对象。

```
        /// <summary>
        /// Delete all of the objects stored in an existing Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket from which the
        /// contents will be deleted.</param>
        /// <returns>A boolean value that represents the success or failure of
        /// deleting all of the objects in the bucket.</returns>
        public static async Task<bool> DeleteBucketContentsAsync(IAmazonS3
client, string bucketName)
        {
            // Iterate over the contents of the bucket and delete all objects.
            var request = new ListObjectsV2Request
            {
                BucketName = bucketName,
            };

            try
            {
                ListObjectsV2Response response;
```

```
                do
                {
                    response = await client.ListObjectsV2Async(request);
                    response.S3Objects
                        .ForEach(async obj => await
client.DeleteObjectAsync(bucketName, obj.Key));

                    // If the response is truncated, set the request
ContinuationToken
                    // from the NextContinuationToken property of the response.
                    request.ContinuationToken = response.NextContinuationToken;
                }
                while (response.IsTruncated);

                return true;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error deleting objects: {ex.Message}");
                return false;
            }
        }
```

删除不受版本控制的 S3 存储桶中的多个对象。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to delete multiple objects from an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class DeleteMultipleObjects
    {
        /// <summary>
        /// The Main method initializes the Amazon S3 client and the name of
        /// the bucket and then passes those values to MultiObjectDeleteAsync.
```

```
        /// </summary>
        public static async Task Main()
        {
            const string bucketName = "doc-example-bucket";

            // If the Amazon S3 bucket from which you wish to delete objects is
not
            // located in the same AWS Region as the default user, define the
            // AWS Region for the Amazon S3 bucket as a parameter to the client
            // constructor.
            IAmazonS3 s3Client = new AmazonS3Client();

            await MultiObjectDeleteAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method uses the passed Amazon S3 client to first create and then
        /// delete three files from the named bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// Amazon S3 methods.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where
objects
        /// will be created and then deleted.</param>
        public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
        {
            // Create three sample objects which we will then delete.
            var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

            // Now perform the multi-object delete, passing the key names and
            // version IDs. Since we are working with a non-versioned bucket,
            // the object keys collection includes null version IDs.
            DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keysAndVersions,
            };

            // You can add a specific object key to the delete request using the
            // AddKey method of the multiObjectDeleteRequest.
            try
```

```
            {
                DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException e)
            {
                PrintDeletionErrorStatus(e);
            }
        }

        /// <summary>
        /// Prints the list of errors raised by the call to DeleteObjectsAsync.
        /// </summary>
        /// <param name="ex">A collection of exceptions returned by the call to
        /// DeleteObjectsAsync.</param>
        public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
        {
            DeleteObjectsResponse errorResponse = ex.Response;
            Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

            Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
            Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

            Console.WriteLine("Printing error data...");
            foreach (DeleteError deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// This method creates simple text file objects that can be used in
        /// the delete method.
        /// </summary>
        /// <param name="client">The Amazon S3 client used to call
PutObjectAsync.</param>
        /// <param name="number">The number of objects to create.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created.</param>
```

```
        /// <returns>A list of keys (object keys) and versions that the calling
        /// method will use to delete the newly created files.</returns>
        public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, int number, string bucketName)
        {
            List<KeyVersion> keys = new List<KeyVersion>();
            for (int i = 0; i < number; i++)
            {
                string key = "ExampleObject-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                PutObjectResponse response = await
client.PutObjectAsync(request);

                // For non-versioned bucket operations, we only need the
                // object key.
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                };
                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

删除受版本控制的 S3 存储桶中的多个对象。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
```

```
   /// This example shows how to delete objects in a version-enabled Amazon
   /// Simple StorageService (Amazon S3) bucket.
   /// </summary>
   public class DeleteMultipleObjects
   {
       public static async Task Main()
       {
           string bucketName = "doc-example-bucket";

           // If the AWS Region for your Amazon S3 bucket is different from
           // the AWS Region of the default user, define the AWS Region for
           // the Amazon S3 bucket and pass it to the client constructor
           // like this:
           // RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
           IAmazonS3 s3Client;

           s3Client = new AmazonS3Client();
           await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
       }

       /// <summary>
       /// This method removes multiple versions and objects from a
       /// version-enabled Amazon S3 bucket.
       /// </summary>
       /// <param name="client">The initialized Amazon S3 client object used to
call
       /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
       /// RemoveDeleteMarkersAsync.</param>
       /// <param name="bucketName">The name of the bucket from which to delete
       /// objects.</param>
       public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
       {
           // Delete objects (specifying object version in the request).
           await DeleteObjectVersionsAsync(client, bucketName);

           // Delete objects (without specifying object version in the request).
           var deletedObjects = await DeleteObjectsAsync(client, bucketName);

           // Additional exercise - remove the delete markers Amazon S3 returned
from
           // the preceding response. This results in the objects reappearing
```

```
            // in the bucket (you can verify the appearance/disappearance of
            // objects in the console).
            await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
        }

        /// <summary>
        /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
        /// them again. The method returns a list of the Amazon S3 objects
deleted.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created and then deleted.</param>
        /// <returns>A list of DeletedObjects.</returns>
        public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

            // Delete objects using only keys. Amazon S3 creates a delete marker
and
            // returns its version ID in the response.
            List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
            return deletedObjects;
        }

        /// <summary>
        /// This method creates several temporary objects and then deletes them.
        /// </summary>
        /// <param name="client">The S3 client.</param>
        /// <param name="bucketName">Name of the bucket.</param>
        /// <returns>Async task.</returns>
        public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

            // Delete the specific object versions.
```

```
            await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
        }

        /// <summary>
        /// Displays the list of information about deleted files to the console.
        /// </summary>
        /// <param name="e">Error information from the delete process.</param>
        private static void DisplayDeletionErrors(DeleteObjectsException e)
        {
            var errorResponse = e.Response;
            Console.WriteLine($"No. of objects successfully deleted =
 {errorResponse.DeletedObjects.Count}");
            Console.WriteLine($"No. of objects failed to delete =
 {errorResponse.DeleteErrors.Count}");
            Console.WriteLine("Printing error data...");
            foreach (var deleteError in errorResponse.DeleteErrors)
            {
                Console.WriteLine($"Object Key:
 {deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
            }
        }

        /// <summary>
        /// Delete multiple objects from a version-enabled bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        private static async Task VersionedDeleteAsync(IAmazonS3 client, string
 bucketName, List<KeyVersion> keys)
        {
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keys, // This includes the object keys and specific
 version IDs.
            };

            try
```

```
            {
                Console.WriteLine("Executing VersionedDelete...");
                DeleteObjectsResponse response = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted all the
 {response.DeletedObjects.Count} items");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Deletes multiple objects from a non-versioned Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="keys">A list of key names for the objects to delete.</
param>
        /// <returns>A list of the deleted objects.</returns>
        private static async Task<List<DeletedObject>>
 NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
 keys)
        {
            // Create a request that includes only the object key names.
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest();
            multiObjectDeleteRequest.BucketName = bucketName;

            foreach (var key in keys)
            {
                multiObjectDeleteRequest.AddKey(key.Key);
            }

            // Execute DeleteObjectsAsync.
            // The DeleteObjectsAsync method adds a delete marker for each
            // object deleted. You can verify that the objects were removed
            // using the Amazon S3 console.
            DeleteObjectsResponse response;
```

```
            try
            {
                Console.WriteLine("Executing NonVersionedDelete...");
                response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
                throw; // Some deletions failed. Investigate before continuing.
            }

            // This response contains the DeletedObjects list which we use to
delete the delete markers.
            return response.DeletedObjects;
        }

        /// <summary>
        /// Deletes the markers left after deleting the temporary objects.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
        /// RemoveDeleteMarkersAsync.</param>
        /// <param name="bucketName">The name of the bucket from which to delete
        /// objects.</param>
        /// <param name="deletedObjects">A list of the objects that were
deleted.</param>
        private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
        {
            var keyVersionList = new List<KeyVersion>();

            foreach (var deletedObject in deletedObjects)
            {
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = deletedObject.Key,
                    VersionId = deletedObject.DeleteMarkerVersionId,
                };
                keyVersionList.Add(keyVersion);
            }
```

```
            // Create another request to delete the delete markers.
            var multiObjectDeleteRequest = new DeleteObjectsRequest
            {
                BucketName = bucketName,
                Objects = keyVersionList,
            };

            // Now, delete the delete marker to bring your objects back to the
 bucket.
            try
            {
                Console.WriteLine("Removing the delete markers .....");
                var deleteObjectResponse = await
 client.DeleteObjectsAsync(multiObjectDeleteRequest);
                Console.WriteLine($"Successfully deleted the
 {deleteObjectResponse.DeletedObjects.Count} delete markers");
            }
            catch (DeleteObjectsException ex)
            {
                DisplayDeletionErrors(ex);
            }
        }

        /// <summary>
        /// Create temporary Amazon S3 objects to show how object deletion wors
 in an
        /// Amazon S3 bucket with versioning enabled.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// PutObjectAsync to create temporary objects for the example.</param>
        /// <param name="bucketName">A string representing the name of the S3
        /// bucket where we will create the temporary objects.</param>
        /// <param name="number">The number of temporary objects to create.</
param>
        /// <returns>A list of the KeyVersion objects.</returns>
        private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
 client, string bucketName, int number)
        {
            var keys = new List<KeyVersion>();

            for (var i = 0; i < number; i++)
            {
```

```
                string key = "ObjectToDelete-" + new System.Random().Next();
                PutObjectRequest request = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = key,
                    ContentBody = "This is the content body!",
                };

                var response = await client.PutObjectAsync(request);
                KeyVersion keyVersion = new KeyVersion
                {
                    Key = key,
                    VersionId = response.VersionId,
                };

                keys.Add(keyVersion);
            }

            return keys;
        }
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for .NET API 参考中的 DeleteObjects。

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
```

```
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket.
#       $2 - A list of keys in the bucket to delete.

# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function delete_items_in_bucket() {
  local bucket_name=$1
  local keys=$2
  local response

  # Create the JSON for the items to delete.
  local delete_items
  delete_items="{\"Objects\":["
  for key in $keys; do
    delete_items="$delete_items{\"Key\": \"$key\"},"
  done
  delete_items=${delete_items%?} # Remove the final comma.
  delete_items="$delete_items]}"

  response=$(aws s3api delete-objects \
    --bucket "$bucket_name" \
    --delete "$delete_items")

  # shellcheck disable=SC2181
  if [[ $? -ne 0 ]]; then
    errecho "ERROR:  AWS reports s3api delete-object operation failed.\n
$response"
    return 1
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteObjects。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

 deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
            client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }
```

```
        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅 AWS SDK for C++ API 参考中的 DeleteObjects。

CLI

AWS CLI

以下命令从名为 my-bucket 的存储桶中删除对象：

```
aws s3api delete-objects --bucket my-bucket --delete file://delete.json
```

delete.json 是当前目录中指定要删除的对象的 JSON 文档：

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

输出：

```
{
    "Deleted": [
        {
            "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUPO1koHwzU",
            "Key": "test1.txt",
            "DeleteMarker": true
        }
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteObjects。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
```

```
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
     log.Printf("Bucket %s does not exist.\n", bucket)
     err = noBucket
    }
   } else if len(delOut.Errors) > 0 {
    for _, outErr := range delOut.Errors {
     log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
    }
    err = fmt.Errorf("%s", *delOut.Errors[0].Message)
   }
  }
  return err
}
```

- 有关 API 详细信息，请参阅 AWS SDK for Go API 参考中的 DeleteObjects。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName>

                Where:
                   bucketName - the Amazon S3 bucket name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                    .key(keyName)
                    .build();
```

```
            putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
                .objects(keys)
                .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
 DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(del)
                    .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 DeleteObjects。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

删除多个对象。

```
import { DeleteObjectsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new DeleteObjectsCommand({
    Bucket: "test-bucket",
    Delete: {
      Objects: [{ Key: "object1.txt" }, { Key: "object2.txt" }],
    },
  });

  try {
    const { Deleted } = await client.send(command);
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
 objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 详细信息，请参阅 AWS SDK for JavaScript API 参考中的 DeleteObjects。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 DeleteObjects。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

从键列表中删除一组对象。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $objects = [];
            foreach ($contents['Contents'] as $content) {
                $objects[] = [
                    'Key' => $content['Key'],
                ];
            }
            $this->s3client->deleteObjects([
                'Bucket' => $this->bucketName,
                'Delete' => [
                    'Objects' => $objects,
                ],
            ]);
            $check = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            if (count($check) <= 0) {
                throw new Exception("Bucket wasn't empty.");
            }
            echo "Deleted all objects and folders from $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with object deletion before continuing.");
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for PHP API 参考中的 DeleteObjects。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令从存储桶“test-files”中移除对象“sample.txt”。在命令执行之前，系统会提示您进行确认；要取消提示，请使用 -Force 开关。

```
Remove-S3Object -BucketName test-files -Key sample.txt
```

示例 2：假设存储桶已配置为启用对象版本，则此命令会从存储桶“test-files”中移除对象“sample.txt”的指定版本。

```
Remove-S3Object -BucketName test-files -Key sample.txt -VersionId
 HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

示例 3：此命令通过单个批量操作，从存储桶“test-files”中移除对象“sample1.txt”、“sample2.txt”和“sample3.txt”。无论删除的成功或错误状态如何，服务响应都将列出所有已处理的键。要仅获取服务无法处理的键的错误，请添加 -ReportErrorsOnly 参数（也可以使用别名 -Quiet 来指定此参数）。

```
Remove-S3Object -BucketName test-files -KeyCollection @( "sample1.txt",
 "sample2.txt", "sample3.txt" )
```

示例 4：此示例使用带有 -KeyCollection 参数的内联表达式来获取要删除的对象的键。Get-S3Object 返回 Amazon.S3.Model.S3Object 实例的集合，每个实例都有一个标识对象的字符串类型的 Key（键）成员。

```
Remove-S3Object -bucketname "test-files" -KeyCollection (Get-S3Object "test-
files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

示例 5：此示例获取存储桶中所有具有键前缀“prefix/subprefix”的对象并将其删除。请注意，一次只能处理一个传入的对象。对于大型集合，可以考虑将集合传递给 cmdlet 的 -InputObject（别名 -S3ObjectCollection）参数，这样，只需调用一次服务即可批量删除。

```
Get-S3Object -BucketName "test-files" -KeyPrefix "prefix/subprefix" | Remove-
S3Object -Force
```

示例 6：此示例将代表删除标记的 Amazon.S3.Model.S3ObjectVersion 实例的集合传送到 cmdlet 进行删除。请注意，一次只能处理一个传入的对象。对于大型集合，可以考虑将集合传

递给 cmdlet 的 -InputObject（别名 -S3ObjectCollection）参数，这样，只需调用一次服务即可批量删除。

```
(Get-S3Version -BucketName "test-files").Versions | Where {$_.IsDeleteMarker -eq
 "True"} | Remove-S3Object -Force
```

示例 7：此脚本显示如何通过构造与 -KeyAndVersionCollection 参数一起使用的对象数组，来批量删除一组对象（在本例中为删除标记）。

```
$keyVersions = @()
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
 {$_.IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
 $marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeleteObjects。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用对象键列表删除一组对象。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
```

```
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3
Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                 and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                    bucket.name,
                )
        except ClientError:
            logger.exception("Couldn't delete any objects from bucket %s.",
bucket.name)
            raise
        else:
            return response
```

删除桶中的所有对象。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise
```

通过删除版本控制对象的所有版本永久删除该对象。

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
```

```
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考>中的 DeleteObjects。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
 # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
 #
 # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
 def delete_bucket(bucket)
   puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")
   answer = gets.chomp.downcase
   if answer == "y"
     bucket.objects.batch_delete!
     bucket.delete
     puts("Emptied and deleted bucket #{bucket.name}.\n")
   end
 rescue Aws::Errors::ServiceError => e
   puts("Couldn't empty and delete bucket #{bucket.name}.")
   puts("\t#{e.code}: #{e.message}")
   raise
 end
```

- 有关 API 详细信息，请参阅 AWS SDK for Ruby API 参考中的 DeleteObjects。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
pub async fn delete_objects(client: &Client, bucket_name: &str) ->
 Result<Vec<String>, Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build()
            .map_err(Error::from)?;
        delete_objects.push(obj_id);
    }

    let return_keys = delete_objects.iter().map(|o| o.key.clone()).collect();

    if !delete_objects.is_empty() {
        client
            .delete_objects()
            .bucket(bucket_name)
            .delete(
                Delete::builder()
                    .set_objects(Some(delete_objects))
                    .build()
                    .map_err(Error::from)?,
            )
            .send()
            .await?;
    }
```

```
    let objects: ListObjectsV2Output =
 client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(Error::unhandled(
            "There were still objects left in the bucket.",
        )),
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 DeleteObjects。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    public func deleteObjects(bucket: String, keys: [String]) async throws {
        let input = DeleteObjectsInput(
            bucket: bucket,
            delete: S3ClientTypes.Delete(
                objects: keys.map({ S3ClientTypes.ObjectIdentifier(key: $0) }),
                quiet: true
            )
        )
```

```
        do {
            let output = try await client.deleteObjects(input: input)

            // As of the last update to this example, any errors are returned
            // in the `output` object's `errors` property. If there are any
            // errors in this array, throw an exception. Once the error
            // handling is finalized in later updates to the AWS SDK for
            // Swift, this example will be updated to handle errors better.

            guard let errors = output.errors else {
                return  // No errors.
            }
            if errors.count != 0 {
                throw ServiceHandlerError.deleteObjectsError
            }
        } catch {
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 DeleteObjects。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeletePublicAccessBlock** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeletePublicAccessBlock。

CLI

AWS CLI

删除存储桶的屏蔽公共访问权限配置

以下 delete-public-access-block 示例移除指定存储桶上的屏蔽公共访问权限配置。

```
aws s3api delete-public-access-block \
    --bucket my-bucket
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeletePublicAccessBlock。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令关闭给定存储桶的屏蔽公共访问权限配置。

```
Remove-S3PublicAccessBlock -BucketName 's3testbucket' -Force -Select
 '^BucketName'
```

输出：

```
s3testbucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 DeletePublicAccessBlock。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketAccelerateConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketAccelerateConfiguration。

CLI

AWS CLI

检索存储桶的加速配置

以下 get-bucket-accelerate-configuration 示例检索指定存储桶的加速配置。

```
aws s3api get-bucket-accelerate-configuration \
    --bucket my-bucket
```

输出：

```
{
```

```
    "Status": "Enabled"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketAccelerateConfiguration。

PowerShell

适用于 PowerShell 的工具

示例 1：如果为指定的存储桶启用了传输加速设置，则此命令将返回值 Enabled。

```
Get-S3BucketAccelerateConfiguration -BucketName 's3testbucket'
```

输出：

```
Value
-----
Enabled
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketAccelerateConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketAcl** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketAcl。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理访问控制列表（ACL）

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Get the access control list (ACL) for the new bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to get the
        /// access control list (ACL) of the bucket.</param>
        /// <param name="newBucketName">The name of the newly created bucket.</
param>
        /// <returns>An S3AccessControlList.</returns>
        public static async Task<S3AccessControlList>
 GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
        {
            // Retrieve bucket ACL to show that the ACL was properly applied to
            // the new bucket.
            GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = newBucketName,
            });

            return getACLResponse.AccessControlList;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 GetBucketAcl。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
            s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
                outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                          << getGranteeTypeString(grantee.GetType()) <<
 std::endl;
            }
```

```
            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                          << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                          << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:            "
                          << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:           "
                          << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                      getPermissionString(grant.GetPermission()) <<
                      std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
*/

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
```

```
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                   "objects in this bucket, and read/write this "
                   "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 GetBucketAcl。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶的访问控制列表：

```
aws s3api get-bucket-acl --bucket my-bucket
```

输出：

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        }
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketAcl。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <objectKey>

                Where:
                  bucketName - The Amazon S3 bucket to get the access control
 list (ACL) for.
                  objectKey - The object to get the ACL for.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getBucketACL(s3, objectKey, bucketName);
```

```
        s3.close();
        System.out.println("Done!");
    }

    public static String getBucketACL(S3Client s3, String objectKey, String
 bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
 grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 GetBucketAcl。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

获取 ACL 权限。

```
import { GetBucketAclCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketAclCommand({
    Bucket: "test-bucket",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 GetBucketAcl。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
```

```
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
acl.owner
            )
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 GetBucketAcl。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketAnalyticsConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketAnalyticsConfiguration。

CLI

AWS CLI

检索具有特定 ID 的存储桶的分析配置

以下 get-bucket-analytics-configuration 示例显示了指定存储桶和 ID 的分析配置。

```
aws s3api get-bucket-analytics-configuration \
    --bucket my-bucket \
    --id 1
```

输出：

```
{
    "AnalyticsConfiguration": {
        "StorageClassAnalysis": {},
        "Id": "1"
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketAnalyticsConfiguration。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回给定 S3 存储桶中名为“testfilter”的分析筛选条件的详细信息。

```
Get-S3BucketAnalyticsConfiguration -BucketName 's3testbucket' -AnalyticsId
 'testfilter'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketAnalyticsConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketCors** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketCors。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Retrieve the CORS configuration applied to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to retrieve the CORS configuration.</param>
        /// <returns>The created CORS configuration object.</returns>
        private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
        {
            GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
            {
                BucketName = BucketName,
            };
            var response = await client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
            return configuration;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 GetBucketCors。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶的跨源资源共享配置：

```
aws s3api get-bucket-cors --bucket my-bucket
```

输出：

```
{
    "CORSRules": [
        {
            "AllowedHeaders": [
                "*"
            ],
            "ExposeHeaders": [
                "x-amz-server-side-encryption"
            ],
            "AllowedMethods": [
                "PUT",
                "POST",
                "DELETE"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedOrigins": [
                "http://www.example.com"
            ]
        },
        {
            "AllowedHeaders": [
                "Authorization"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedMethods": [
                "GET"
            ],
            "AllowedOrigins": [
                "*"
            ]
        }
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketCors。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

获取存储桶的 CORS 策略。

```
import { GetBucketCorsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketCorsCommand({
    Bucket: "test-bucket",
  });

  try {
    const { CORSRules } = await client.send(command);
    CORSRules.forEach((cr, i) => {
      console.log(
        `\nCORSRule ${i + 1}`,
        `\n${"-".repeat(10)}`,
        `\nAllowedHeaders: ${cr.AllowedHeaders.join(" ")}`,
        `\nAllowedMethods: ${cr.AllowedMethods.join(" ")}`,
        `\nAllowedOrigins: ${cr.AllowedOrigins.join(" ")}`,
        `\nExposeHeaders: ${cr.ExposeHeaders.join(" ")}`,
        `\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考>中的 GetBucketCors。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_cors(self):
        """
        Get the CORS rules for the bucket.

        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception(("Couldn't get CORS for bucket %s.",
 self.bucket.name))
            raise
        else:
            return cors
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 GetBucketCors。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
 configuration for the bucket.
  def get_cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 GetBucketCors。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketEncryption** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketEncryption。

CLI

AWS CLI

检索存储桶的服务器端加密配置

以下 get-bucket-encryption 示例检索存储桶 my-bucket 的服务器端加密配置。

```
aws s3api get-bucket-encryption \
    --bucket my-bucket
```

输出：

```
{
    "ServerSideEncryptionConfiguration": {
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                }
            }
        ]
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketEncryption。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回与给定存储桶关联的所有服务器端加密规则。

```
Get-S3BucketEncryption -BucketName 's3casetestbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketEncryption。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 GetBucketInventoryConfiguration 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketInventoryConfiguration。

CLI

AWS CLI

检索存储桶的清单配置

以下 get-bucket-inventory-configuration 示例检索 ID 为 1 的指定存储桶的清单配置。

```
aws s3api get-bucket-inventory-configuration \
    --bucket my-bucket \
    --id 1
```

输出：

```
{
    "InventoryConfiguration": {
        "IsEnabled": true,
        "Destination": {
            "S3BucketDestination": {
                "Format": "ORC",
                "Bucket": "arn:aws:s3:::my-bucket",
                "AccountId": "123456789012"
            }
        },
        "IncludedObjectVersions": "Current",
        "Id": "1",
        "Schedule": {
```

```
            "Frequency": "Weekly"
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketInventoryConfiguration。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回给定 S3 存储桶的名为“testinventory”的清单的详细信息。

```
Get-S3BucketInventoryConfiguration -BucketName 's3testbucket' -InventoryId
 'testinventory'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketInventoryConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketLifecycleConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketLifecycleConfiguration。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Returns a configuration object for the supplied bucket name.
```

```
        /// </summary>
        /// <param name="client">The S3 client object used to call
        /// the GetLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">The name of the S3 bucket for which a
        /// configuration will be created.</param>
        /// <returns>Returns a new LifecycleConfiguration object.</returns>
        public static async Task<LifecycleConfiguration>
 RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
        {
            var request = new GetLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 GetBucketLifecycleConfiguration。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶的生命周期配置：

```
aws s3api get-bucket-lifecycle-configuration --bucket my-bucket
```

输出：

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
```

```
                    "StorageClass": "GLACIER"
                }
            ]
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 0,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketLifecycleConfiguration。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
        else:
            return config.rules
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 GetBucketLifecycleConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketLocation** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketLocation。

CLI

AWS CLI

如果存在约束条件，则以下命令会检索名为 my-bucket 的存储桶的位置约束：

```
aws s3api get-bucket-location --bucket my-bucket
```

输出：

```
{
    "LocationConstraint": "us-west-2"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketLocation。

PowerShell

适用于 PowerShell 的工具

示例 1：如果存在约束，则此命令返回存储桶“s3testbucket”的位置约束。

```
Get-S3BucketLocation -BucketName 's3testbucket'
```

输出：

```
Value
-----
ap-south-1
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketLocation。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(),
 Error> {
    let resp = client.list_buckets().send().await?;
    let buckets = resp.buckets();
```

```
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint().unwrap().as_ref() == region {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 GetBucketLocation。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketLogging** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketLogging。

CLI

AWS CLI

检索存储桶的日志记录状态

以下 get-bucket-logging 示例检索指定存储桶的日志记录状态。

```
aws s3api get-bucket-logging \
    --bucket my-bucket
```

输出：

```
{
    "LoggingEnabled": {
        "TargetPrefix": "",
        "TargetBucket": "my-bucket-logs"
          }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketLogging。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回指定存储桶的日志记录状态。

```
Get-S3BucketLogging -BucketName 's3testbucket'
```

输出：

```
TargetBucketName   Grants TargetPrefix
----------------   ------ ------------
testbucket1        {}     testprefix
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketLogging。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketMetricsConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketMetricsConfiguration。

CLI

AWS CLI

检索具有特定 ID 的存储桶的指标配置

以下 get-bucket-metrics-configuration 示例显示了指定存储桶和 ID 的指标配置。

```
aws s3api get-bucket-metrics-configuration \
    --bucket my-bucket \
    --id 123
```

输出：

```
{
    "MetricsConfiguration": {
        "Filter": {
            "Prefix": "logs"
        },
        "Id": "123"
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketMetricsConfiguration。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回有关给定 S3 存储桶的名为“testfilter”的指标筛选条件的详细信息。

```
Get-S3BucketMetricsConfiguration -BucketName 's3testbucket' -MetricsId
 'testfilter'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketMetricsConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketNotification** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketNotification。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶的通知配置：

```
aws s3api get-bucket-notification --bucket my-bucket
```

输出：

```
{
    "TopicConfiguration": {
        "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
        "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
        "Event": "s3:ObjectCreated:*",
        "Events": [
            "s3:ObjectCreated:*"
        ]
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketNotification。

PowerShell

适用于 PowerShell 的工具

示例 1：此示例检索给定存储桶的通知配置

```
Get-S3BucketNotification -BucketName kt-tools | select -ExpandProperty
 TopicConfigurations
```

输出：

```
Id   Topic
--   -----
mimo arn:aws:sns:eu-west-1:123456789012:topic-1
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketNotification。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketPolicy** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketPolicy。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
            s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
```

```
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
 <<
                  policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关更多信息，请参阅《AWS SDK for C++ API 参考》中的 GetBucketPolicy。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶的存储桶策略：

```
aws s3api get-bucket-policy --bucket my-bucket
```

输出：

```
{
    "Policy": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Sid\":\"\",\"Effect
\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":
\"arn:aws:s3:::my-bucket/*\"},{\"Sid\":\"\",\"Effect\":\"Deny\",\"Principal\":
\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::my-bucket/secret/*
\"}]}"
}
```

获取并放置存储桶策略 以下示例演示了如何下载 Amazon S3 存储桶策略，修改文件，然后使用 put-bucket-policy 来应用修改后的存储桶策略。要将存储桶策略下载到文件中，您可以运行：

```
aws s3api get-bucket-policy --bucket mybucket --query Policy --output text >
 policy.json
```

然后，您可以根据需要修改 policy.json 文件。最后，您可以通过运行以下对象，将此修改后的策略应用回 S3 存储桶：

policy.json 文件（根据需要）。最后，您可以通过运行以下对象，将此修改后的策略应用回 S3 存储桶：

文件（根据需要）。最后，您可以通过运行以下对象，将此修改后的策略应用回 S3 存储桶：

```
aws s3api put-bucket-policy --bucket mybucket --policy file://policy.json
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketPolicy。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to get the policy from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }

        return "";
    }
}
```

- 有关更多信息，请参阅《AWS SDK for Java 2.x API 参考》中的 GetBucketPolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

获取存储桶策略。

```
import { GetBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketPolicyCommand({
    Bucket: "test-bucket",
  });

  try {
    const { Policy } = await client.send(command);
    console.log(JSON.parse(Policy));
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关更多信息，请参阅《AWS SDK for JavaScript API 参考》中的 GetBucketPolicy。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request =
        GetBucketPolicyRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 GetBucketPolicy。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令输出与给定 S3 存储桶关联的存储桶策略。

```
Get-S3BucketPolicy -BucketName 's3testbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketPolicy。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
 self.bucket.name)
            raise
        else:
            return json.loads(policy.policy)
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 GetBucketPolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def get_policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- 有关更多信息，请参阅《AWS SDK for Ruby API 参考》中的 GetBucketPolicy。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketPolicyStatus** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketPolicyStatus。

CLI

AWS CLI

检索存储桶的策略状态，此状态指示存储桶是否为公有存储桶

以下 get-bucket-policy-status 示例检索存储桶 my-bucket 的策略状态。

```
aws s3api get-bucket-policy-status \
    --bucket my-bucket
```

输出：

```
{
    "PolicyStatus": {
        "IsPublic": false
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketPolicyStatus。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回给定 S3 存储桶的策略状态，此状态指示存储桶是否为公有存储桶。

```
Get-S3BucketPolicyStatus -BucketName 's3casetestbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketPolicyStatus。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketReplication** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketReplication。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶的复制配置：

```
aws s3api get-bucket-replication --bucket my-bucket
```

输出：

```
{
    "ReplicationConfiguration": {
        "Rules": [
            {
                "Status": "Enabled",
                "Prefix": "",
                "Destination": {
                    "Bucket": "arn:aws:s3:::my-bucket-backup",
                    "StorageClass": "STANDARD"
                },
                "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
            }
        ],
        "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketReplication。

PowerShell

适用于 PowerShell 的工具

示例 1：返回在名为“mybucket”的存储桶上设置的复制配置信息。

```
Get-S3BucketReplication -BucketName mybucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketReplication。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 GetBucketRequestPayment 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketRequestPayment。

CLI

AWS CLI

检索存储桶的请求付款配置

以下 get-bucket-request-payment 示例检索指定存储桶的申请方付款配置。

```
aws s3api get-bucket-request-payment \
    --bucket my-bucket
```

输出：

```
{
    "Payer": "BucketOwner"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketRequestPayment。

PowerShell

适用于 PowerShell 的工具

示例 1：返回名为“mybucket”的存储桶的请求付款配置。默认情况下，存储桶拥有者支付从存储桶进行下载的费用。

```
Get-S3BucketRequestPayment -BucketName mybucket
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketRequestPayment。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketTagging** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketTagging。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶的标记配置：

```
aws s3api get-bucket-tagging --bucket my-bucket
```

输出：

```
{
    "TagSet": [
        {
            "Value": "marketing",
            "Key": "organization"
        }
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketTagging。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回与给定存储桶关联的所有标签。

```
Get-S3BucketTagging -BucketName 's3casetestbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketTagging。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketVersioning** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketVersioning。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶的版本控制配置：

```
aws s3api get-bucket-versioning --bucket my-bucket
```

输出：

```
{
    "Status": "Enabled"
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketVersioning。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回与给定存储桶相关的版本控制状态。

```
Get-S3BucketVersioning -BucketName 's3testbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketVersioning。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetBucketWebsite** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetBucketWebsite。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
                // Get the website configuration.
                GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
                Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- 有关更多信息，请参阅《AWS SDK for .NET API 参考》中的 GetBucketWebsite。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
```

```
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
            s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                  << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
 outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
                  << "Error page: "
                  << websiteResult.GetErrorDocument().GetKey()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关更多信息，请参阅《AWS SDK for C++ API 参考》中的 GetBucketWebsite。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶的静态网站配置：

```
aws s3api get-bucket-website --bucket my-bucket
```

输出：

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetBucketWebsite。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

获取网站配置。

```
import { GetBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetBucketWebsiteCommand({
    Bucket: "test-bucket",
  });

  try {
    const { ErrorDocument, IndexDocument } = await client.send(command);
    console.log(
      `Your bucket is set up to host a website. It has an error document:`,
      `${ErrorDocument.Key}, and an index document: ${IndexDocument.Suffix}.`,
    );
  } catch (err) {
```

```
    console.error(err);
  }
};
```

- 有关更多信息，请参阅《AWS SDK for JavaScript API 参考》中的 GetBucketWebsite。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回给定 S3 存储桶的静态网站配置的详细信息。

```
Get-S3BucketWebsite -BucketName 's3testbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetBucketWebsite。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基础知识
- 如果对象已修改，则从桶中获取该对象
- 从多区域接入点中获取对象
- 加密入门
- 跟踪上传和下载操作

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Shows how to download an object from an Amazon S3 bucket to the
        /// local computer.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket where the object is
        /// currently stored.</param>
        /// <param name="objectName">The name of the object to download.</param>
        /// <param name="filePath">The path, including filename, where the
        /// downloaded object will be stored.</param>
        /// <returns>A boolean value indicating the success or failure of the
        /// download process.</returns>
        public static async Task<bool> DownloadObjectFromBucketAsync(
            IAmazonS3 client,
            string bucketName,
            string objectName,
            string filePath)
        {
            // Create a GetObject request
            var request = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
            };

            // Issue request and remember to dispose of the response
            using GetObjectResponse response = await
client.GetObjectAsync(request);

            try
            {
```

```
                // Save object to local file
                await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
                return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error saving {objectName}: {ex.Message}");
                return false;
            }
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 GetObject。

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#       $1 - The name of the bucket to download the object from.
```

```
#       $2 - The path and file name to store the downloaded bucket.
#       $3 - The key (name) of the object in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function download_object_from_bucket() {
  local bucket_name=$1
  local destination_file_name=$2
  local object_name=$3
  local response

  response=$(aws s3api get-object \
    --bucket "$bucket_name" \
    --key "$object_name" \
    "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetObject。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
```

```
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
            client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
                  err.GetExceptionName() << ": " << err.GetMessage() <<
 std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
                  << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 GetObject。

CLI

AWS CLI

以下示例使用 `get-object` 命令从 Amazon S3 下载对象：

```
aws s3api get-object --bucket text-content --key dir/
my_images.tar.bz2 my_images.tar.bz2
```

请注意，指定 outfile 参数时没有诸如“--outfile”之类的选项名称。输出文件的名称必须是命令中的最后一个参数。

以下示例演示了如何使用 `--range` 从对象下载特定字节范围。请注意，字节范围需要以“bytes=”为前缀：

```
aws s3api get-object --bucket text-content --key dir/my_data --
range bytes=8888-9999 my_data_range
```

有关检索对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“获取对象”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetObject。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(bucketName string, objectKey string,
 fileName string) error {
 result, err := basics.S3Client.GetObject(context.TODO(), &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
 objectKey, err)
  return err
 }
 defer result.Body.Close()
 file, err := os.Create(fileName)
 if err != nil {
```

```
  log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
  return err
 }
 defer file.Close()
 body, err := io.ReadAll(result.Body)
 if err != nil {
  log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
 err)
 }
 _, err = file.Write(body)
 return err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 GetObject。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用 S3Client 以字节数组读取数据。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <path>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
```

```
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

使用 S3TransferManager 将 S3 存储桶中的对象下载到本地文件中。查看完整文件并进行测试。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

```
   public Long downloadFile(S3TransferManager transferManager, String
 bucketName,
                            String key, String downloadedFileWithPath) {
       DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
               .getObjectRequest(b -> b.bucket(bucketName).key(key))
               .destination(Paths.get(downloadedFileWithPath))
               .build();

       FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);

       CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
       logger.info("Content length [{}]",
 downloadResult.response().contentLength());
       return downloadResult.response().contentLength();
   }
```

使用 S3Client 读取属于对象的标签。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    public static void listTags(S3Client s3, String bucketName, String keyName) {
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            GetObjectTaggingResponse tags =
s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

使用 S3Client 获取对象的 URL。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
```

```
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

使用 S3Client 通过 S3Presigner 客户端对象获取对象。

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents a text file.\s
                """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
                .region(region)
                .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
 String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();
```

```
            GetObjectPresignRequest getObjectPresignRequest =
 GetObjectPresignRequest.builder()
                    .signatureDuration(Duration.ofMinutes(60))
                    .getObjectRequest(getObjectRequest)
                    .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
 presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
 presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
 values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
            });

            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                        OutputStream httpOutputStream =
 connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.getStackTrace();
        }
    }
}
```

使用 ResponseTransformer 对象和 S3Client 获取对象。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <path>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                    path - The path where the file is written to.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
 keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                    .builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 GetObject。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

下载对象。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new GetObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
  });

  try {
    const response = await client.send(command);
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
 methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 GetObject。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 GetObject。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

获取对象。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $file = $this->s3client->getObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
            ]);
            $body = $file->get('Body');
            $body->rewind();
            echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
        } catch (Exception $exception) {
            echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
            exit("Please fix error with file downloading before continuing.");
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for PHP API 参考中的 GetObject。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令从存储桶“test-files”中检索项目“sample.txt”，并将其保存到当前位置名为“local-sample.txt”的文件中。在调用此命令之前，文件“local-sample.txt”不必存在。

```
Read-S3Object -BucketName test-files -Key sample.txt -File local-sample.txt
```

示例 2：此命令从存储桶“test-files”中检索虚拟目录“DIR”，并将其保存到当前位置名为“Local-DIR”的文件夹中。在调用此命令之前，文件夹“Local-DIR”不必存在。

```
Read-S3Object -BucketName test-files -KeyPrefix DIR -Folder Local-DIR
```

示例 3：将键以“.json”结尾的所有对象从存储桶名称中带有“config”的存储桶下载到指定文件夹中的文件。对象键用于设置文件名。

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -
like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetObject。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get(self):
        """
        Gets the object.
```

```
        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        else:
            return body
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 GetObject。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

获取对象。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
```

```
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is
 downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes} downloaded to
 #{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

获取对象并报告其服务器端加密状态。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
```

```
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? "no" :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 GetObject。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, anyhow::Error> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone())?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await? {
        let bytes_len = bytes.len();
        file.write_all(&bytes)?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }

    Ok(byte_count)
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 GetObject。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   TRY.
       oo_result = lo_s3->getobject(           " oo_result is returned for
testing purposes. "
                  iv_bucket = iv_bucket_name
                  iv_key = iv_object_key
                ).
       DATA(lv_object_data) = oo_result->get_body( ).
       MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
     CATCH /aws1/cx_s3_nosuchbucket.
       MESSAGE 'Bucket does not exist.' TYPE 'E'.
     CATCH /aws1/cx_s3_nosuchkey.
       MESSAGE 'Object key does not exist.' TYPE 'E'.
   ENDTRY.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API 参考》中的 GetObject。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

将对象从桶下载到本地文件。

```
    public func downloadFile(bucket: String, key: String, to: String) async
throws {
        let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the data stream object. Return immediately if there isn't one.
        guard let body = output.body,
              let data = try await body.readData() else {
            return
        }
        try data.write(to: fileUrl)
    }
```

将对象读入到 Swift 数据对象。

```
    public func readFile(bucket: String, key: String) async throws -> Data {
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        let output = try await client.getObject(input: input)

        // Get the stream and return its contents in a `Data` object. If
        // there is no stream, return an empty `Data` object instead.
        guard let body = output.body,
              let data = try await body.readData() else {
            return "".data(using: .utf8)!
        }
```

```
        return data
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 GetObject。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectAcl** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObjectAcl。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理访问控制列表（ACL）

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
            s3Client.GetObjectAcl(request);
```

```
   if (!outcome.IsSuccess()) {
       const Aws::S3::S3Error &err = outcome.GetError();
       std::cerr << "Error: getObjectAcl: "
                 << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
   } else {
       Aws::Vector<Aws::S3::Model::Grant> grants =
               outcome.GetResult().GetGrants();

       for (auto it = grants.begin(); it != grants.end(); it++) {
           std::cout << "For object " << objectKey << ": "
                     << std::endl << std::endl;

           Aws::S3::Model::Grant grant = *it;
           Aws::S3::Model::Grantee grantee = grant.GetGrantee();

           if (grantee.TypeHasBeenSet()) {
               std::cout << "Type:          "
                         << getGranteeTypeString(grantee.GetType()) <<
std::endl;
           }

           if (grantee.DisplayNameHasBeenSet()) {
               std::cout << "Display name:  "
                         << grantee.GetDisplayName() << std::endl;
           }

           if (grantee.EmailAddressHasBeenSet()) {
               std::cout << "Email address: "
                         << grantee.GetEmailAddress() << std::endl;
           }

           if (grantee.IDHasBeenSet()) {
               std::cout << "ID:            "
                         << grantee.GetID() << std::endl;
           }

           if (grantee.URIHasBeenSet()) {
               std::cout << "URI:           "
                         << grantee.GetURI() << std::endl;
           }

           std::cout << "Permission:    " <<
```

```
                        getPermissionString(grant.GetPermission()) <<
                        std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
 string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string
*/
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                   "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can read this object's data and its metadata";
```

```
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this object's permissions";
        default:
            return "Permission unknown";
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 GetObjectAcl。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶中对象的访问控制列表：

```
aws s3api get-object-acl --bucket my-bucket --key index.html
```

输出：

```
{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
```

```
            "Permission": "READ"
        }
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetObjectAcl。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 GetObjectAcl。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
                self.object.key,
                acl.owner["DisplayName"],
            )
        except ClientError:
            logger.exception("Couldn't get ACL for object %s.", self.object.key)
            raise
        else:
            return acl
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 GetObjectAcl。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectAttributes** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObjectAttributes。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 使用 Amazon S3 对象完整性

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of
 the object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
```

```
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
```

```
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
```

```
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object
 parts." <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 GetObjectAttributes。

CLI

AWS CLI

从对象检索元数据而不返回对象本身

以下 get-object-attributes 示例从对象 doc1.rtf 检索元数据。

```
aws s3api get-object-attributes \
    --bucket my-bucket \
    --key doc1.rtf \
    --object-attributes "StorageClass" "ETag" "ObjectSize"
```

输出：

```
{
```

```
    "LastModified": "2022-03-15T19:37:31+00:00",
    "VersionId": "IuCPjXTDzHNfldAuitVBIKJpF2p1fg4P",
    "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
    "StorageClass": "STANDARD",
    "ObjectSize": 405
}
```

有关更多信息，请参阅《Amazon S3 API 参考》中的 GetObjectAttributes。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetObjectAttributes。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectLegalHold** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObjectLegalHold。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 获取对象的法定保留配置
- 锁定 Amazon S3 对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
/// <summary>
/// Get the legal hold details for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
```

```
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
 {bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 GetObjectLegalHold。

CLI

AWS CLI

检索对象的法定保留状态

以下 `get-object-legal-hold` 示例检索指定对象的法定保留状态。

```
aws s3api get-object-legal-hold \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf
```

输出：

```
{
```

```
    "LegalHold": {
        "Status": "ON"
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetObjectLegalHold。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
```

```
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 GetObjectLegalHold。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   // Get the legal hold details for an S3 object.
   public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
       try {
           GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
               .bucket(bucketName)
               .key(objectKey)
               .build();
```

```
            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 GetObjectLegalHold。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

放置对象法定保留。

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
bucket)
```

```
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
            key,
            bucket,
            e,
        )
```

- 有关 API 详细信息，请参阅《适用于 Python 的 AWS SDK（Boto3）API 参考》中的 GetObjectLegalHold。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectLockConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObjectLockConfiguration。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 锁定 Amazon S3 对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   /// <summary>
   /// Get the object lock configuration details for an S3 bucket.
   /// </summary>
   /// <param name="bucketName">The bucket to get details.</param>
   /// <returns>The bucket's object lock configuration details.</returns>
   public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
   {
       try
       {
           var request = new GetObjectLockConfigurationRequest()
           {
               BucketName = bucketName
           };

           var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
           Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                             $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                             $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

           return response.ObjectLockConfiguration;
       }
       catch (AmazonS3Exception ex)
       {
           Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
           return new ObjectLockConfiguration();
       }
```

```
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 GetObjectLockConfiguration。

CLI

AWS CLI

检索存储桶的对象锁定配置

以下 `get-object-lock-configuration` 示例检索指定存储桶的对象锁定配置。

```
aws s3api get-object-lock-configuration \
    --bucket my-bucket-with-object-lock
```

输出：

```
{
    "ObjectLockConfiguration": {
        "ObjectLockEnabled": "Enabled",
        "Rule": {
            "DefaultRetention": {
                "Mode": "COMPLIANCE",
                "Days": 50
            }
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetObjectLockConfiguration。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
  lockConfig = output.ObjectLockConfiguration
```

```
 }

 return lockConfig, err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 GetObjectLockConfiguration。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   // Get the object lock configuration details for an S3 bucket.
   public void getBucketObjectLockConfiguration(String bucketName) {
       GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
           .bucket(bucketName)
           .build();

       GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
       System.out.println("Bucket object lock config for "+bucketName +":  ");
       System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
       System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 GetObjectLockConfiguration。

JavaScript

适用于 JavaScript 的 SDK（v3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import { fileURLToPath } from "url";
import {
  GetObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new GetObjectLockConfigurationCommand({
    Bucket: bucketName,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
  });

  try {
    const { ObjectLockConfiguration } = await client.send(command);
    console.log(`Object Lock Configuration: ${ObjectLockConfiguration}`);
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 GetObjectLockConfiguration。

PowerShell

适用于 PowerShell 的工具

示例 1：如果为给定的 S3 存储桶启用了对象锁定配置，则此命令将返回值“Enabled”。

```
Get-S3ObjectLockConfiguration -BucketName 's3buckettesting' -Select
 ObjectLockConfiguration.ObjectLockEnabled
```

输出：

```
Value
-----
Enabled
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetObjectLockConfiguration。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

获取对象锁定配置。

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:
    """
    Check if object lock is enabled for a bucket.

    Args:
        s3_client: Boto3 S3 client.
```

```
        bucket: The name of the bucket to check.

    Returns:
        True if object lock is enabled, False otherwise.
    """
    try:
        response = s3_client.get_object_lock_configuration(Bucket=bucket)
        return (
            "ObjectLockConfiguration" in response
            and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
"Enabled"
        )
    except s3_client.exceptions.ClientError as e:
        if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
            return False
        else:
            raise
```

- 有关 API 详细信息，请参阅《适用于 Python 的 AWS SDK（Boto3）API 参考》中的 GetObjectLockConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectRetention** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObjectRetention。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 锁定 Amazon S3 对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 GetObjectRetention。

CLI

AWS CLI

检索对象的对象保留配置

以下 get-object-retention 示例检索指定对象的对象保留配置。

```
aws s3api get-object-retention \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf
```

输出：

```
{
    "Retention": {
        "Mode": "GOVERNANCE",
        "RetainUntilDate": "2025-01-01T00:00:00.000Z"
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetObjectRetention。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
```

```
 S3Manager *manager.Uploader
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 GetObjectRetention。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   // Get the retention period for an S3 object.
   public ObjectLockRetention getObjectRetention(String bucketName, String key){
       try {
           GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

           GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
           System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
           return response.retention();

       } catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           return null;
       }
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 GetObjectRetention。

JavaScript

适用于 JavaScript 的 SDK（v3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import { fileURLToPath } from "url";
import { GetObjectRetentionCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new GetObjectRetentionCommand({
    Bucket: bucketName,
    Key: objectKey,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const { Retention } = await client.send(command);
    console.log(`Object Retention Settings: ${Retention.Status}`);
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 GetObjectRetention。

PowerShell

适用于 PowerShell 的工具

示例 1：该命令返回保留对象之前的模式和日期。

```
Get-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetObjectRetention。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetObjectTagging** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetObjectTagging。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 标签入门

CLI

AWS CLI

检索附加到对象的标签

以下 get-object-tagging 示例从指定的对象中检索指定键的值。

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc1.rtf
```

输出：

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        }
    ]
}
```

以下 get-object-tagging 示例尝试检索没有标签的对象 doc2.rtf 的标签集。

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc2.rtf
```

输出：

```
{
    "TagSet": []
}
```

以下 get-object-tagging 示例检索具有多个标签的对象 doc3.rtf 的标签集。

```
aws s3api get-object-tagging \
    --bucket my-bucket \
    --key doc3.rtf
```

输出：

```
{
    "TagSet": [
        {
            "Value": "confidential",
            "Key": "designation"
        },
        {
            "Value": "finance",
            "Key": "department"
        },
        {
            "Value": "payroll",
```

```
            "Key": "team"
        }
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetObjectTagging。

PowerShell

适用于 PowerShell 的工具

示例 1：该示例返回与给定 S3 存储桶上存在的对象关联的标签。

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'testbucket123'
```

输出：

```
Key  Value
---  -----
test value
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetObjectTagging。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetPublicAccessBlock** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetPublicAccessBlock。

CLI

AWS CLI

设置或修改存储桶的屏蔽公共访问权限配置

以下 get-public-access-block 示例显示了指定存储桶的屏蔽公共访问权限配置。

```
aws s3api get-public-access-block \
    --bucket my-bucket
```

输出：

```
{
    "PublicAccessBlockConfiguration": {
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "BlockPublicAcls": true,
        "RestrictPublicBuckets": true
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetPublicAccessBlock。

PowerShell

适用于 PowerShell 的工具

示例 1：该命令返回给定 S3 存储桶的公共访问权限屏蔽设置。

```
Get-S3PublicAccessBlock -BucketName 's3testbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetPublicAccessBlock。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **HeadBucket** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 HeadBucket。

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
##############################################################################
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#       $1 - The name of the bucket to check.
#
# Returns:
#       0 - If the bucket already exists.
#       1 - If the bucket doesn't exist.
##############################################################################
function bucket_exists() {
  local bucket_name
  bucket_name=$1

  # Check whether the bucket already exists.
  # We suppress all output - we're interested only in the return code.

  if aws s3api head-bucket \
    --bucket "$bucket_name" \
    >/dev/null 2>&1; then
    return 0 # 0 in Bash script means true.
  else
    return 1 # 1 in Bash script means false.
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 HeadBucket。

CLI

AWS CLI

以下命令验证对名为 my-bucket 的存储桶的访问权限：

```
aws s3api head-bucket --bucket my-bucket
```

如果存储桶存在并且您可以访问它，则不返回任何输出。否则，会显示错误消息。例如：

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 HeadBucket。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(bucketName string) (bool, error) {
 _, err := basics.S3Client.HeadBucket(context.TODO(), &s3.HeadBucketInput{
  Bucket: aws.String(bucketName),
 })
 exists := true
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NotFound:
    log.Printf("Bucket %v is available.\n", bucketName)
    exists = false
    err = nil
```

```
   default:
    log.Printf("Either you don't have access to bucket %v or another error
 occurred. "+
     "Here's what happened: %v\n", bucketName, err)
   }
  }
 } else {
  log.Printf("Bucket %v exists and you already own it.", bucketName)
 }

 return exists, err
}
```

- 有关 API 详细，请参阅《AWS SDK for Go API 参考》中的 HeadBucket。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def exists(self):
        """
```

```
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name,
            )
            exists = False
        return exists
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 HeadBucket。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **HeadObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 HeadObject。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶中对象的元数据：

```
aws s3api head-object --bucket my-bucket --key index.html
```

输出：

```
{
    "AcceptRanges": "bytes",
    "ContentType": "text/html",
    "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
    "ContentLength": 77,
```

```
    "VersionId": "null",
    "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
    "Metadata": {}
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 HeadObject。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

确定对象的内容类型。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>>

                Where:
```

```
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                    .key(keyName)
                    .bucket(bucketName)
                    .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

获取对象的还原状态。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName>\s

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - A key name that represents the object.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    public static void checkStatus(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(keyName)
                    .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
 response.restore());
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细，请参阅《AWS SDK for Java 2.x API 参考》中的 HeadObject。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
 #{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
    false
  end
```

```
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细，请参阅《AWS SDK for Ruby API 参考》中的 HeadObject。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListBucketAnalyticsConfigurations** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListBucketAnalyticsConfigurations。

CLI

AWS CLI

检索存储桶的分析配置列表

下面的 list-bucket-analytics-configurations 检索指定存储桶的分析配置列表。

```
aws s3api list-bucket-analytics-configurations \
    --bucket my-bucket
```

输出：

```
{
    "AnalyticsConfigurationList": [
        {
            "StorageClassAnalysis": {},
```

```
            "Id": "1"
        }
    ],
    "IsTruncated": false
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListBucketAnalyticsConfigurations。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回给定 S3 存储桶的前 100 个分析配置。

```
Get-S3BucketAnalyticsConfigurationList -BucketName 's3casetestbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 ListBucketAnalyticsConfigurations。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListBucketInventoryConfigurations** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListBucketInventoryConfigurations。

CLI

AWS CLI

检索存储桶的清单配置列表

以下 list-bucket-inventory-configurations 示例列出了指定存储桶的清单配置。

```
aws s3api list-bucket-inventory-configurations \
    --bucket my-bucket
```

输出：

```
{
```

```
    "InventoryConfigurationList": [
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "ORC",
                    "Bucket": "arn:aws:s3:::my-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "1",
            "Schedule": {
                "Frequency": "Weekly"
            }
        },
        {
            "IsEnabled": true,
            "Destination": {
                "S3BucketDestination": {
                    "Format": "CSV",
                    "Bucket": "arn:aws:s3:::my-bucket",
                    "AccountId": "123456789012"
                }
            },
            "IncludedObjectVersions": "Current",
            "Id": "2",
            "Schedule": {
                "Frequency": "Daily"
            }
        }
    ],
    "IsTruncated": false
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListBucketInventoryConfigurations。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回给定 S3 存储桶的前 100 个清单配置。

```
Get-S3BucketInventoryConfigurationList -BucketName 's3testbucket'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 ListBucketInventoryConfigurations。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListBuckets** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListBuckets。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
    public class ListBuckets
    {
        private static IAmazonS3 _s3Client;

        /// <summary>
        /// Get a list of the buckets owned by the default user.
```

```
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <returns>The response from the ListingBuckets call that contains a
        /// list of the buckets owned by the default user.</returns>
        public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
 client)
        {
            return await client.ListBucketsAsync();
        }

        /// <summary>
        /// This method lists the name and creation date for the buckets in
        /// the passed List of S3 buckets.
        /// </summary>
        /// <param name="bucketList">A List of S3 bucket objects.</param>
        public static void DisplayBucketList(List<S3Bucket> bucketList)
        {
            bucketList
                .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
 created on: {b.CreationDate}"));
        }

        public static async Task Main()
        {
            // The client uses the AWS Region of the default user.
            // If the Region where the buckets were created is different,
            // pass the Region to the client constructor. For example:
            // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
            _s3Client = new AmazonS3Client();
            var response = await GetBuckets(_s3Client);
            DisplayBucketList(response.Buckets);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 ListBuckets。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
 buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 ListBuckets。

CLI

AWS CLI

以下命令使用 `list-buckets` 命令显示所有 Amazon S3 存储桶的名称（跨所有区域）：

```
aws s3api list-buckets --query "Buckets[].Name"
```

查询选项会筛选 `list-buckets` 的输出，使其范围缩小到仅限存储桶名称。

有关存储桶的更多信息，请参阅《Amazon S3 开发人员指南》中的“使用 Amazon S3 存储桶”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListBuckets。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets() ([]types.Bucket, error) {
 result, err := basics.S3Client.ListBuckets(context.TODO(),
 &s3.ListBucketsInput{})
 var buckets []types.Bucket
 if err != nil {
  log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
 } else {
  buckets = result.Buckets
 }
 return buckets, err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 ListBuckets。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }
    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
        List<Bucket> bucketList = response.buckets();
```

```
        for (Bucket bucket: bucketList) {
            System.out.println("Bucket name "+bucket.name());
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 ListBuckets。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

列出存储桶。

```
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new ListBucketsCommand({});

  try {
    const { Owner, Buckets } = await client.send(command);
    console.log(
      `${Owner.DisplayName} owns ${Buckets.length} bucket${
        Buckets.length === 1 ? "" : "s"
      }:`,
    );
    console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 ListBuckets。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令返回所有 S3 存储桶。

```
Get-S3Bucket
```

示例 2：此命令返回名为“test-files”的存储桶

```
Get-S3Bucket -BucketName test-files
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 ListBuckets。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
Boto3
                            that contains collections and factory methods to
create
                            other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
            return buckets
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 ListBuckets。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource
```

```
  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts "Found these buckets:"
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 ListBuckets。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(),
 Error> {
    let resp = client.list_buckets().send().await?;
    let buckets = resp.buckets();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint().unwrap().as_ref() == region {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 ListBuckets。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
/// Return an array containing information about every available bucket.
///
/// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
///   each bucket.
public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
    let output = try await client.listBuckets(input: ListBucketsInput())

    guard let buckets = output.buckets else {
        return []
    }
    return buckets
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 ListBuckets。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListMultipartUploads** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListMultipartUploads。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 删除未完成的分段上传

CLI

AWS CLI

以下命令列出了名为 my-bucket 的存储桶的所有活动分段上传：

```
aws s3api list-multipart-uploads --bucket my-bucket
```

输出：

```
{
    "Uploads": [
        {
            "Initiator": {
                "DisplayName": "username",
                "ID": "arn:aws:iam::0123456789012:user/username"
            },
            "Initiated": "2015-06-02T18:01:30.000Z",
            "UploadId":
 "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
            "StorageClass": "STANDARD",
            "Key": "multipart/01",
            "Owner": {
                "DisplayName": "aws-account-name",
                "ID":
 "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
            }
        }
    ],
    "CommonPrefixes": []
}
```

正在进行的分段上传会产生 Amazon S3 存储费用。完成或中止活动分段上传，可将其从您的账户中移除。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListMultipartUploads。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key()
 + "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 ListMultipartUploads。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListObjectVersions** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListObjectVersions。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 处理版本控制对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example lists the versions of the objects in a version enabled
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class ListObjectVersions
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket";

        // If the AWS Region where your bucket is defined is different from
        // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
        // for the AWS Region to the client constructor like this:
        //      var client = new AmazonS3Client(RegionEndpoint.USWest2);
        IAmazonS3 client = new AmazonS3Client();
        await GetObjectListWithAllVersionsAsync(client, bucketName);
    }

    /// <summary>
    /// This method lists all versions of the objects within an Amazon S3
```

```
        /// version enabled bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// ListVersionsAsync.</param>
        /// <param name="bucketName">The name of the version enabled Amazon S3
 bucket
        /// for which you want to list the versions of the contained objects.</
param>
        public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
 client, string bucketName)
        {
            try
            {
                // When you instantiate the ListVersionRequest, you can
                // optionally specify a key name prefix in the request
                // if you want a list of object versions of a specific object.

                // For this example we set a small limit in MaxKeys to return
                // a small list of versions.
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    MaxKeys = 2,
                };

                do
                {
                    ListVersionsResponse response = await
 client.ListVersionsAsync(request);

                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine($"key: {entry.Key} size:
 {entry.Size}");
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
```

```
                    else
                    {
                        request = null;
                    }
                }
                while (request != null);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}'");
            }
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 ListObjectVersions。

CLI

AWS CLI

以下命令检索名为 my-bucket 的存储桶中对象的版本信息：

```
aws s3api list-object-versions --bucket my-bucket --prefix index.html
```

输出：

```
{
    "DeleteMarkers": [
        {
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": true,
            "VersionId": "B2VsEK5saUNNHKcOAJj7hIE86RozToyq",
            "Key": "index.html",
            "LastModified": "2015-11-10T00:57:03.000Z"
        },
        {
```

```
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
            "Key": "index.html",
            "LastModified": "2015-11-09T23:32:20.000Z"
        }
    ],
    "Versions": [
        {
            "LastModified": "2015-11-10T00:20:11.000Z",
            "VersionId": "Rb_l2T8UHDkFEwCgJjhlgPOZC0qJ.vpD",
            "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T23:26:41.000Z",
            "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAFf",
            "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 38
        },
        {
            "LastModified": "2015-11-09T22:50:50.000Z",
            "VersionId": "null",
            "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
```

```
            "StorageClass": "STANDARD",
            "Key": "index.html",
            "Owner": {
                "DisplayName": "my-username",
                "ID":
 "7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
            },
            "IsLatest": false,
            "Size": 533823
        }
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListObjectVersions。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
```

```
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 ListObjectVersions。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 ListObjectVersions。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `ListObjects` 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListObjects`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 创建列出 Amazon S3 对象的网页

CLI

AWS CLI

以下示例使用 `list-objects` 命令显示指定存储桶中所有对象的名称：

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

该示例使用 `--query` 参数筛选 `list-objects` 的输出，使其范围缩小到每个对象的键值和大小

有关对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“使用 Amazon S3 对象”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListObjects。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令检索有关存储桶“test-files”中所有项目的信息。

```
Get-S3Object -BucketName test-files
```

示例 2：此命令从存储桶“test-files”中检索有关项目“sample.txt”的信息。

```
Get-S3Object -BucketName test-files -Key sample.txt
```

示例 3：此命令从存储桶“test-files”中检索有关前缀为“sample”的所有项目的信息。

```
Get-S3Object -BucketName test-files -KeyPrefix sample
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 ListObjects。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListObjectsV2** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListObjectsV2。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基础知识

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Shows how to list the objects in an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the bucket for which to list
        /// the contents.</param>
        /// <returns>A boolean value indicating the success or failure of the
        /// copy operation.</returns>
```

```
        public static async Task<bool> ListBucketContentsAsync(IAmazonS3 client,
 string bucketName)
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 5,
                };

                Console.WriteLine("--------------------------------------");
                Console.WriteLine($"Listing the contents of {bucketName}:");
                Console.WriteLine("--------------------------------------");

                ListObjectsV2Response response;

                do
                {
                    response = await client.ListObjectsV2Async(request);

                    response.S3Objects
                        .ForEach(obj => Console.WriteLine($"{obj.Key,-35}
{obj.LastModified.ToShortDateString(),10}{obj.Size,10}"));

                    // If the response is truncated, set the request
 ContinuationToken
                    // from the NextContinuationToken property of the response.
                    request.ContinuationToken = response.NextContinuationToken;
                }
                while (response.IsTruncated);

                return true;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error encountered on server.
 Message:'{ex.Message}' getting list of objects.");
                return false;
            }
        }
```

使用分页工具列出对象。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// The following example lists objects in an Amazon Simple Storage
    /// Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectsPaginator
    {
        private const string BucketName = "doc-example-bucket";

        public static async Task Main()
        {
            IAmazonS3 s3Client = new AmazonS3Client();

            Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");
            await ListingObjectsAsync(s3Client, BucketName);
        }

        /// <summary>
        /// This method uses a paginator to retrieve the list of objects in an
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">An Amazon S3 client object.</param>
        /// <param name="bucketName">The name of the S3 bucket whose objects
        /// you want to list.</param>
        public static async Task ListingObjectsAsync(IAmazonS3 client, string
 bucketName)
        {
            var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
 ListObjectsV2Request
            {
                BucketName = bucketName,
            });

            await foreach (var response in listObjectsV2Paginator.Responses)
            {
                Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
                Console.WriteLine($"Number of Keys: {response.KeyCount}");
```

```
                foreach (var entry in response.S3Objects)
                {
                    Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
                }
            }
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 ListObjectsV2。

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#       $1 - The name of the bucket.
#
# Returns:
```

```
#       The list of files in text format.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListObjectsV2。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
 {
    Aws::S3::S3Client s3Client(clientConfig);
```

```
    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                    outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << "  " << object.GetKey() << std::endl;
        keysResult.push_back(object.GetKey());
    }

    return true;
}
```

- 有关 API 详细信息，请参阅 AWS SDK for C++ API 参考中的 ListObjectsV2。

CLI

AWS CLI

获取存储桶中对象的列表

以下 `list-objects-v2` 示例列出了指定存储桶中的对象。

```
aws s3api list-objects-v2 \
    --bucket my-bucket
```

输出：

```
{
    "Contents": [
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"621503c373607d548b37cff8778d992c\"",
            "StorageClass": "STANDARD",
            "Key": "doc1.rtf",
            "Size": 391
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",
            "StorageClass": "STANDARD",
            "Key": "doc2.rtf",
            "Size": 373
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"08210852f65a2e9cb999972539a64d68\"",
            "StorageClass": "STANDARD",
            "Key": "doc3.rtf",
            "Size": 399
        },
        {
            "LastModified": "2019-11-05T23:11:50.000Z",
            "ETag": "\"d1852dd683f404306569471af106988e\"",
            "StorageClass": "STANDARD",
            "Key": "doc4.rtf",
            "Size": 6225
        }
```

```
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListObjectsV2。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(bucketName string) ([]types.Object, error)
 {
 result, err := basics.S3Client.ListObjectsV2(context.TODO(),
 &s3.ListObjectsV2Input{
  Bucket: aws.String(bucketName),
 })
 var contents []types.Object
 if err != nil {
  log.Printf("Couldn't list objects in bucket %v. Here's why: %v\n", bucketName,
 err)
 } else {
  contents = result.Contents
 }
```

```
 return contents, err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 ListObjectsV2。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s
```

```
                Where:
                    bucketName - The Amazon S3 bucket from which objects are
read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                System.out.print("\n The object is " + calKb(myValue.size()) + "
KBs");
                System.out.print("\n The owner is " + myValue.owner());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // convert bytes to kbs.
    private static long calKb(Long val) {
```

```
        return val / 1024;
    }
}
```

使用分页列出对象。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The Amazon S3 bucket from which objects are
 read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
```

```
                    .bucket(bucketName)
                    .maxKeys(1)
                    .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                    .flatMap(r -> r.contents().stream())
                    .forEach(content -> System.out.println(" Key: " +
 content.key() + " size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 ListObjectsV2。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

列出存储桶中的所有对象。如果有多个对象，则将使用 IsTruncated 和 NextContinuationToken 来迭代完整列表。

```
import {
  S3Client,
  // This command supersedes the ListObjectsCommand and is the recommended way to
 list objects.
  ListObjectsV2Command,
} from "@aws-sdk/client-s3";

const client = new S3Client({});
```

```
export const main = async () => {
  const command = new ListObjectsV2Command({
    Bucket: "my-bucket",
    // The default and maximum number of keys returned is 1000. This limits it to
    // one for demonstration purposes.
    MaxKeys: 1,
  });

  try {
    let isTruncated = true;

    console.log("Your bucket contains the following objects:\n");
    let contents = "";

    while (isTruncated) {
      const { Contents, IsTruncated, NextContinuationToken } =
        await client.send(command);
      const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
      contents += contentsList + "\n";
      isTruncated = IsTruncated;
      command.input.ContinuationToken = NextContinuationToken;
    }
    console.log(contents);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 ListObjectsV2。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun listBucketObjects(bucketName: String) {
```

```
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 ListObjectsV2。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

列出存储桶中的对象。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        try {
            $contents = $this->s3client->listObjectsV2([
                'Bucket' => $this->bucketName,
            ]);
            echo "The contents of your bucket are: \n";
            foreach ($contents['Contents'] as $content) {
                echo $content['Key'] . "\n";
            }
```

```
        } catch (Exception $exception) {
            echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
            exit("Please fix error with listing objects before continuing.");
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for PHP API 参考》中的 ListObjectsV2。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
are listed.
        :return: The list of objects.
        """
```

```
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
 bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
 bucket.name)
            raise
        else:
            return objects
```

- 有关 API 详细信息，请参阅《AWS SDK for Python（Boto3）API 参考》中的 ListObjectsV2。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end
```

```
  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
 #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 ListObjectsV2。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
pub async fn list_objects(client: &Client, bucket: &str) -> Result<(), Error> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 ListObjectsV2。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    TRY.
        oo_result = lo_s3->listobjectsv2(         " oo_result is returned for
testing purposes. "
          iv_bucket = iv_bucket_name
        ).
        MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API 参考》中的 ListObjectsV2。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    public func listBucketFiles(bucket: String) async throws -> [String] {
```

```
        let input = ListObjectsV2Input(
            bucket: bucket
        )
        let output = try await client.listObjectsV2(input: input)
        var names: [String] = []

        guard let objList = output.contents else {
            return []
        }

        for obj in objList {
            if let objName = obj.key {
                names.append(objName)
            }
        }

        return names
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 ListObjectsV2。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketAccelerateConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketAccelerateConfiguration。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.Threading.Tasks;
```

```
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
    /// bucket-level feature that enables you to perform faster data transfers
    /// to Amazon S3. This example shows how to configure Transfer
    /// Acceleration.
    /// </summary>
    public class TransferAcceleration
    {
        /// <summary>
        /// The main method initializes the client object and sets the
        /// Amazon Simple Storage Service (Amazon S3) bucket name before
        /// calling EnableAccelerationAsync.
        /// </summary>
        public static async Task Main()
        {
            var s3Client = new AmazonS3Client();
            const string bucketName = "doc-example-bucket";

            await EnableAccelerationAsync(s3Client, bucketName);
        }

        /// <summary>
        /// This method sets the configuration to enable transfer acceleration
        /// for the bucket referred to in the bucketName parameter.
        /// </summary>
        /// <param name="client">An Amazon S3 client used to enable the
        /// acceleration on an Amazon S3 bucket.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
the
        /// method will be enabling acceleration.</param>
        private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
        {
            try
            {
                var putRequest = new PutBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                    AccelerateConfiguration = new AccelerateConfiguration
                    {
                        Status = BucketAccelerateStatus.Enabled,
```

```
                    },
                };
                await client.PutBucketAccelerateConfigurationAsync(putRequest);

                var getRequest = new GetBucketAccelerateConfigurationRequest
                {
                    BucketName = bucketName,
                };
                var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

                Console.WriteLine($"Acceleration state = '{response.Status}' ");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
            }
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 PutBucketAccelerateConfiguration。

CLI

AWS CLI

设置存储桶的加速配置

以下 put-bucket-accelerate-configuration 示例启用指定存储桶的加速配置。

```
aws s3api put-bucket-accelerate-configuration \
    --bucket my-bucket \
    --accelerate-configuration Status=Enabled
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketAccelerateConfiguration。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令为给定 S3 存储桶启用传输加速。

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')
Write-S3BucketAccelerateConfiguration -BucketName 's3testbucket' -
AccelerateConfiguration_Status $statusVal
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutBucketAccelerateConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketAcl** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketAcl。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理访问控制列表（ACL）

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Creates an Amazon S3 bucket with an ACL to control access to the
        /// bucket and the objects stored in it.
```

```
        /// </summary>
        /// <param name="client">The initialized client object used to create
        /// an Amazon S3 bucket, with an ACL applied to the bucket.
        /// </param>
        /// <param name="region">The AWS Region where the bucket will be
created.</param>
        /// <param name="newBucketName">The name of the bucket to create.</param>
        /// <returns>A boolean value indicating success or failure.</returns>
        public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
        {
            try
            {
                // Create a new Amazon S3 bucket with Canned ACL.
                var putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = region,
                    CannedACL = S3CannedACL.LogDeliveryWrite,
                };

                PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

                return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Amazon S3 error: {ex.Message}");
            }

            return false;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 PutBucketAcl。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
 &ownerID,
                              const Aws::String &granteePermission,
                              const Aws::String &granteeType, const Aws::String
 &granteeID,
                              const Aws::String &granteeEmailAddress,
                              const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));
```

```
    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
            s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
*/

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
```

```
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
*/

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 PutBucketAcl。

CLI

AWS CLI

此示例向两个 AWS 用户（user1@example.com 和user2@example.com）授予 `full control` 权限，并向所有人授予 `read` 权限：

```
aws s3api put-bucket-acl --bucket MyBucket --grant-full-
control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-
read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

有关自定义 ACL（s3api ACL 命令（如 `put-bucket-acl`）使用相同的速记参数表示法）的详细信息，请参阅 http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketAcl。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <id>\s

                Where:
                  bucketName - The Amazon S3 bucket to grant permissions on.\s
                  id - The ID of the owner of this bucket (you can get this value
 from the AWS Management Console).
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                    .grantee(builder -> builder.id(id)
                            .type(Type.CANONICAL_USER))
                    .permission(Permission.FULL_CONTROL)
                    .build();

            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                    .owner(builder -> builder.id(id))
                    .grants(grantList2)
                    .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                    .bucket(bucketName)
                    .accessControlPolicy(acl)
                    .build();

            s3.putBucketAcl(putAclReq);
```

```
        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 PutBucketAcl。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

放置存储桶 ACL。

```
import { PutBucketAclCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Most Amazon S3 use cases don't require the use of access control lists (ACLs).
// We recommend that you disable ACLs, except in unusual circumstances where
// you need to control access for each object individually.
// Consider a policy instead. For more information see https://
docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html.
export const main = async () => {
  // Grant a user READ access to a bucket.
  const command = new PutBucketAclCommand({
    Bucket: "test-bucket",
    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
            // The canonical ID of the user. This ID is an obfuscated form of
 your AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
```

```
            // For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/finding-canonical-user-id.html.
            ID: "canonical-id-1",
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "FULL_CONTROL",
        },
      ],
      Owner: {
        ID: "canonical-id-2",
      },
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 PutBucketAcl。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun setBucketAcl(
    bucketName: String,
```

```
    idVal: String,
) {
    val myGrant =
        Grantee {
            id = idVal
            type = Type.CanonicalUser
        }

    val ownerGrant =
        Grant {
            grantee = myGrant
            permission = Permission.FullControl
        }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb =
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerOb
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 PutBucketAcl。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
 recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
                    "Permission": "WRITE",
```

```
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 PutBucketAcl。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketCors** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketCors。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Add CORS configuration to the Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to apply the CORS configuration to an Amazon S3 bucket.</param>
        /// <param name="configuration">The CORS configuration to apply.</param>
        private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
        {
```

```
            PutCORSConfigurationRequest request = new
 PutCORSConfigurationRequest()
            {
                BucketName = BucketName,
                Configuration = configuration,
            };

            _ = await client.PutCORSConfigurationAsync(request);
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 PutBucketCors。

CLI

AWS CLI

以下示例启用来自 www.example.com 的 PUT、POST 和 DELETE 请求，并启用来自任何域的 GET 请求：

```
aws s3api put-bucket-cors --bucket MyBucket --cors-configuration file://cors.json

cors.json:
{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketCors。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <accountId>\s

                Where:
                    bucketName - The Amazon S3 bucket to upload an object into.
```

```
                    accountId - The id of the account that owns the Amazon S3
bucket.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    public static void deleteBucketCorsInformation(S3Client s3, String
bucketName, String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
```

```
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                    .allowedMethods(allowMethods)
                    .allowedOrigins(allowOrigins)
                    .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                    .corsRules(corsRules)
                    .build();

            PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
                    .bucket(bucketName)
```

```
                    .corsConfiguration(configuration)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 PutBucketCors。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

添加 CORS 规则。

```
import { PutBucketCorsCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// By default, Amazon S3 doesn't allow cross-origin requests. Use this command
// to explicitly allow cross-origin requests.
export const main = async () => {
  const command = new PutBucketCorsCommand({
    Bucket: "test-bucket",
    CORSConfiguration: {
      CORSRules: [
        {
          // Allow all headers to be sent to this bucket.
          AllowedHeaders: ["*"],
```

```
          // Allow only GET and PUT methods to be sent to this bucket.
          AllowedMethods: ["GET", "PUT"],
          // Allow only requests from the specified origin.
          AllowedOrigins: ["https://www.example.com"],
          // Allow the entity tag (ETag) header to be returned in the response.
 The ETag header
          // The entity tag represents a specific version of the object. The ETag
 reflects
          // changes only to the contents of an object, not its metadata.
          ExposeHeaders: ["ETag"],
          // How long the requesting browser should cache the preflight response.
 After
          // this time, the preflight request will have to be made again.
          MaxAgeSeconds: 3600,
        },
      ],
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 PutBucketCors。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
 are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
 self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
 self.bucket.name)
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 PutBucketCors。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
 an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
```

```
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 PutBucketCors。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketEncryption** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketEncryption。

CLI

AWS CLI

配置存储桶的服务器端加密

以下 put-bucket-encryption 示例将 AES256 加密设置为指定存储桶的默认值。

```
aws s3api put-bucket-encryption \
    --bucket my-bucket \
    --server-side-encryption-configuration '{"Rules":
 [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketEncryption。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令在指定的存储桶上启用具有 Amazon S3 托管式密钥的默认 AES256 服务器端加密（SSE-S3）。

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =
 @{ServerSideEncryptionAlgorithm = "AES256"}}
Set-S3BucketEncryption -BucketName 's3testbucket' -
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutBucketEncryption。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketLifecycleConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketLifecycleConfiguration。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 删除未完成的分段上传
- 处理版本控制对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        /// <summary>
        /// Adds lifecycle configuration information to the S3 bucket named in
        /// the bucketName parameter.
        /// </summary>
        /// <param name="client">The S3 client used to call the
        /// PutLifecycleConfigurationAsync method.</param>
        /// <param name="bucketName">A string representing the S3 bucket to
```

```
        /// which configuration information will be added.</param>
        /// <param name="configuration">A LifecycleConfiguration object that
        /// will be applied to the S3 bucket.</param>
        public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
 string bucketName, LifecycleConfiguration configuration)
        {
            var request = new PutLifecycleConfigurationRequest()
            {
                BucketName = bucketName,
                Configuration = configuration,
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 PutBucketLifecycleConfiguration。

CLI

AWS CLI

以下命令将生命周期配置应用于名为 my-bucket 的存储桶：

```
aws s3api put-bucket-lifecycle-configuration --bucket my-bucket --lifecycle-
configuration  file://lifecycle.json
```

文件 lifecycle.json 是当前文件夹中指定两个规则的 JSON 文档：

```
{
    "Rules": [
        {
            "ID": "Move rotated logs to Glacier",
            "Prefix": "rotated/",
            "Status": "Enabled",
            "Transitions": [
                {
                    "Date": "2015-11-10T00:00:00.000Z",
                    "StorageClass": "GLACIER"
                }
            ]
```

```
        },
        {
            "Status": "Enabled",
            "Prefix": "",
            "NoncurrentVersionTransitions": [
                {
                    "NoncurrentDays": 2,
                    "StorageClass": "GLACIER"
                }
            ],
            "ID": "Move old versions to Glacier"
        }
    ]
}
```

第一条规则在指定日期将带有前缀 `rotated` 的文件移动到 Glacier。第二条规则在旧对象版本不再是最新版本时将其移至 Glacier。有关可接受时间戳格式的信息，请参阅《AWS CLI 用户指南》中的“指定参数值”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketLifecycleConfiguration。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
```

```
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class LifecycleConfiguration {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                  <bucketName> <accountId>\s

                                Where:
                                  bucketName - The Amazon Simple Storage Service
 (Amazon S3) bucket to upload an object into.
                                  accountId - The id of the account that owns the
 Amazon S3 bucket.
                                """;

                if (args.length != 2) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String bucketName = args[0];
                String accountId = args[1];
                Region region = Region.US_EAST_1;
                S3Client s3 = S3Client.builder()
                                .region(region)
                                .build();
```

```
                setLifecycleConfig(s3, bucketName, accountId);
                getLifecycleConfig(s3, bucketName, accountId);
                deleteLifecycleConfig(s3, bucketName, accountId);
                System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
                s3.close();
        }

        public static void setLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
                        // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
                        // S3 Glacier.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("glacierobjects/")
                                        .build();

                        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(0)
                                        .build();

                        LifecycleRule rule1 = LifecycleRule.builder()
                                        .id("Archive immediately rule")
                                        .filter(ruleFilter)
                                        .transitions(transition)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Create a second rule.
                        Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(0)
                                        .build();

                        List<Transition> transitionList = new ArrayList<>();
                        transitionList.add(transition2);

                        LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
```

```
                                        .prefix("glacierobjects/")
                                        .build();

                        LifecycleRule rule2 = LifecycleRule.builder()
                                        .id("Archive and then delete rule")
                                        .filter(ruleFilter2)
                                        .transitions(transitionList)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Add the LifecycleRule objects to an ArrayList.
                        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
                        ruleList.add(rule1);
                        ruleList.add(rule2);

                        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                        .rules(ruleList)
                                        .build();

                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
                                        .bucket(bucketName)

.lifecycleConfiguration(lifecycleConfiguration)
                                        .expectedBucketOwner(accountId)
                                        .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Retrieve the configuration and add a new rule.
        public static void getLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
                try {
                        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
```

```
                                        .builder()
                                        .bucket(bucketName)
                                        .expectedBucketOwner(accountId)
                                        .build();

                        GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
                        List<LifecycleRule> newList = new ArrayList<>();
                        List<LifecycleRule> rules = response.rules();
                        for (LifecycleRule rule : rules) {
                                newList.add(rule);
                        }

                        // Add a new rule with both a prefix predicate and a tag
predicate.
                        LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                                        .prefix("YearlyDocuments/")
                                        .build();

                        Transition transition = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
                                        .days(3650)
                                        .build();

                        LifecycleRule rule1 = LifecycleRule.builder()
                                        .id("NewRule")
                                        .filter(ruleFilter)
                                        .transitions(transition)
                                        .status(ExpirationStatus.ENABLED)
                                        .build();

                        // Add the new rule to the list.
                        newList.add(rule1);
                        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                                        .rules(newList)
                                        .build();

                        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                                        .builder()
```

```
                                        .bucket(bucketName)

 .lifecycleConfiguration(lifecycleConfiguration)
                                        .expectedBucketOwner(accountId)
                                        .build();


 s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }

        // Delete the configuration from the Amazon S3 bucket.
        public static void deleteLifecycleConfig(S3Client s3, String bucketName,
 String accountId) {
                try {
                        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest
 = DeleteBucketLifecycleRequest
                                        .builder()
                                        .bucket(bucketName)
                                        .expectedBucketOwner(accountId)
                                        .build();

                        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

                } catch (S3Exception e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 PutBucketLifecycleConfiguration。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
 configuration can
        be used to archive or delete the objects in the bucket according to
 specified
        parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
```

```
        except ClientError:
            logger.exception(
                "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 PutBucketLifecycleConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketLogging** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketLogging。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
```

```
    /// </summary>
    public class ServerAccessLogging
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }
```

```
        /// <summary>
        /// This method grants appropriate permissions for logging to the
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);
```

```
            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
 stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
```

```
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
 local settings.
                .Build();
        }
    }
```

- 有关更多信息，请参阅《AWS SDK for .NET API 参考》中的 PutBucketLogging。

CLI

AWS CLI

示例 1：设置存储桶策略日志记录

以下 put-bucket-logging 示例为 MyBucket 设置了日志记录策略。首先，使用 put-bucket-policy 命令在存储桶策略中向日志记录服务主体授予权限。

```
aws s3api put-bucket-policy \
    --bucket MyBucket \
    --policy file://policy.json
```

policy.json 的内容：

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {"Service": "logging.s3.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::MyBucket/Logs/*",
            "Condition": {
                "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
                "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
            }
        }
    ]
}
```

要应用日志记录策略，请使用 put-bucket-logging。

```
aws s3api put-bucket-logging \
    --bucket MyBucket \
    --bucket-logging-status file://logging.json
```

logging.json 的内容：

```
{
     "LoggingEnabled": {
         "TargetBucket": "MyBucket",
         "TargetPrefix": "Logs/"
     }
 }
```

向日志记录服务主体授予 s3:PutObject 权限需要使用 put-bucket-policy 命令。

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 Amazon S3 服务器访问日志记录。

示例 2：设置仅向单个用户授予日志访问权限的存储桶策略

以下 put-bucket-logging 示例为 MyBucket 设置了日志记录策略。AWS 用户 bob@example.com 将对日志文件具有完全控制权限，其他人没有任何访问权限。首先，使用 put-bucket-acl 授予 S3 权限。

```
aws s3api put-bucket-acl \
```

```
    --bucket MyBucket \
    --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \
    --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

然后，使用 put-bucket-logging 应用日志记录策略。

```
aws s3api put-bucket-logging \
    --bucket MyBucket \
    --bucket-logging-status file://logging.json
```

logging.json 的内容：

```
{
    "LoggingEnabled": {
        "TargetBucket": "MyBucket",
        "TargetPrefix": "MyBucketLogs/",
        "TargetGrants": [
            {
                "Grantee": {
                    "Type": "AmazonCustomerByEmail",
                    "EmailAddress": "bob@example.com"
                },
                "Permission": "FULL_CONTROL"
            }
        ]
    }
}
```

必须使用 put-bucket-acl 命令向 S3 的日志传输系统授予必要的权限（write-acp 和 read-acp 权限）。

有关更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的 Amazon S3 服务器访问日志记录。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketLogging。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketNotification** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketNotification。

CLI

AWS CLI

将通知配置应用于名为 my-bucket 的存储桶：

```
aws s3api put-bucket-notification --bucket my-bucket --notification-
configuration file://notification.json
```

文件 notification.json 是当前文件夹中的一个 JSON 文件，用于指定 SNS 主题和要监控的事件类型：

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

SNS 主题必须附加一个 IAM 策略，以允许 Amazon S3 向其发布：

```
{
 "Version": "2008-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "example-statement-ID",
   "Effect": "Allow",
   "Principal": {
     "Service": "s3.amazonaws.com"
   },
   "Action": [
    "SNS:Publish"
   ],
   "Resource": "arn:aws:sns:us-west-2:123456789012:my-bucket",
   "Condition": {
      "ArnLike": {
      "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"
    }
   }
  }
 ]
```

```
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketNotification。

PowerShell

适用于 PowerShell 的工具

示例 1：此示例为 S3 事件 ObjectRemovedDelete 配置 SNS 主题配置，并为给定的 s3 存储桶启用通知

```
$topic =  [Amazon.S3.Model.TopicConfiguration] @{
  Id = "delete-event"
  Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
  Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName kt-tools -TopicConfiguration $topic
```

示例 2：此示例为给定存储桶启用 ObjectCreatedAll 的通知，将其发送到 Lambda 函数。

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
  Id = "ObjectCreated-Lambda"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".pem"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
 $lambdaConfig
```

示例 3：此示例基于不同的键后缀创建 2 个不同的 Lambda 配置，并在单个命令中配置了这两个配置。

```
#Lambda Config 1
```

```
$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = "s3:ObjectCreated:*"
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
  Id = "ObjectCreated-dada-ps1"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".ps1"}
      )
    }
  }
}

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName ssm-editor -LambdaFunctionConfiguration
 $firstLambdaConfig,$secondlambdaConfig
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutBucketNotification。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketNotificationConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketNotificationConfiguration。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 处理 S3 事件通知
- 向 EventBridge 发送事件通知

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to enable notifications for an Amazon Simple
    /// Storage Service (Amazon S3) bucket.
    /// </summary>
    public class EnableNotifications
    {
        public static async Task Main()
        {
            const string bucketName = "doc-example-bucket1";
            const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
            const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

            IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
```

```
            await EnableNotificationAsync(client, bucketName, snsTopic,
sqsQueue);
        }

        /// <summary>
        /// This method makes the call to the PutBucketNotificationAsync method.
        /// </summary>
        /// <param name="client">An initialized Amazon S3 client used to call
        /// the PutBucketNotificationAsync method.</param>
        /// <param name="bucketName">The name of the bucket for which
        /// notifications will be turned on.</param>
        /// <param name="snsTopic">The ARN for the Amazon Simple Notification
        /// Service (Amazon SNS) topic associated with the S3 bucket.</param>
        /// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
        /// (Amazon SQS) queue to which notifications will be pushed.</param>
        public static async Task EnableNotificationAsync(
            IAmazonS3 client,
            string bucketName,
            string snsTopic,
            string sqsQueue)
        {
            try
            {
                // The bucket for which we are setting up notifications.
                var request = new PutBucketNotificationRequest()
                {
                    BucketName = bucketName,
                };

                // Defines the topic to use when sending a notification.
                var topicConfig = new TopicConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic,
                };
                request.TopicConfigurations = new List<TopicConfiguration>
                {
                    topicConfig,
                };
                request.QueueConfigurations = new List<QueueConfiguration>
                {
                    new QueueConfiguration()
                    {
```

```
                        Events = new List<EventType>
{ EventType.ObjectCreatedPut },
                        Queue = sqsQueue,
                    },
                };

                // Now apply the notification settings to the bucket.
                PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 PutBucketNotificationConfiguration。

CLI

AWS CLI

启用存储桶的指定通知

以下 put-bucket-notification-configuration 示例将通知配置应用于名为 my-bucket 的存储桶。文件 notification.json 是当前文件夹中的一个 JSON 文件，用于指定 SNS 主题和要监控的事件类型。

```
aws s3api put-bucket-notification-configuration \
    --bucket my-bucket \
    --notification-configuration file://notification.json
```

notification.json 的内容：

```
{
    "TopicConfigurations": [
        {
```

```
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
            "Events": [
                "s3:ObjectCreated:*"
            ]
        }
    ]
}
```

SNS 主题必须附加一个 IAM 策略，以允许 Amazon S3 向其发布。

```
{
    "Version": "2008-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:my-bucket"
                }
            }
        }
    ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketNotificationConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketPolicy** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketPolicy。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                 const Aws::String &policyBody,
                                 const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
            Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
            s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}
```

```
//! Build a policy JSON string.
/*!
  \param userArn: Aws user Amazon Resource Name (ARN).
      For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
  \param bucketName: Name of a bucket.
  \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                            const Aws::String &bucketName) {
    return
            "{\n"
            "   \"Version\":\"2012-10-17\",\n"
            "   \"Statement\":[\n"
            "       {\n"
            "           \"Sid\": \"1\",\n"
            "           \"Effect\": \"Allow\",\n"
            "           \"Principal\": {\n"
            "               \"AWS\": \""
            + userArn +
            "\"\n""           },\n"
            "           \"Action\": [ \"s3:getObject\" ],\n"
            "           \"Resource\": [ \"arn:aws:s3:::"
            + bucketName +
            "/*\" ]\n"
            "       }\n"
            "   ]\n"
            "}";
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 PutBucketPolicy。

CLI

AWS CLI

此示例允许所有用户检索 MyBucket 中的任何对象，但 MySecretFolder 中的对象除外。它还向 AWS 账户 1234-5678-9012 的根用户授予 put 和 delete 权限：

```
aws s3api put-bucket-policy --bucket MyBucket --policy file://policy.json

policy.json:
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::MyBucket/*"
      },
      {
         "Effect": "Deny",
         "Principal": "*",
         "Action": "s3:GetObject",
         "Resource": "arn:aws:s3:::MyBucket/MySecretFolder/*"
      },
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:root"
         },
         "Action": [
            "s3:DeleteObject",
            "s3:PutObject"
         ],
         "Resource": "arn:aws:s3:::MyBucket/*"
      }

  ]
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketPolicy。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <polFile>

                Where:
                    bucketName - The Amazon S3 bucket to set the policy on.
                    polFile - A JSON file containing the policy (see the Amazon
 S3 Readme for an example).\s
```

```
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    public static void setPolicy(S3Client s3, String bucketName, String
policyText) {
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    // Loads a JSON-formatted policy from a file
```

```
    public static String getBucketPolicyFromFile(String policyFile) {

        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
 StandardCharsets.UTF_8);
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
 ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 PutBucketPolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

添加策略。

```
import { PutBucketPolicyCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutBucketPolicyCommand({
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Sid: "AllowGetObject",
          // Allow this particular user to call GetObject on any object in this
 bucket.
          Effect: "Allow",
          Principal: {
            AWS: "arn:aws:iam::ACCOUNT-ID:user/USERNAME",
          },
          Action: "s3:GetObject",
          Resource: "arn:aws:s3:::BUCKET-NAME/*",
        },
      ],
    }),
    // Apply the preceding policy to this bucket.
    Bucket: "BUCKET-NAME",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 PutBucketPolicy。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy,
 self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
 self.bucket.name)
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 PutBucketPolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
 configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def set_policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- 有关 API 详细信息，请参阅《AWS SDK for Ruby API 参考》中的 PutBucketPolicy。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketReplication** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketReplication。

CLI

AWS CLI

为 S3 存储桶配置复制

以下 put-bucket-replication 示例将复制配置应用于指定的 S3 存储桶。

```
aws s3api put-bucket-replication \
    --bucket AWSDOC-EXAMPLE-BUCKET1 \
    --replication-configuration file://replication.json
```

replication.json 的内容：

```
{
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role",
    "Rules": [
        {
            "Status": "Enabled",
            "Priority": 1,
            "DeleteMarkerReplication": { "Status": "Disabled" },
            "Filter" : { "Prefix": ""},
            "Destination": {
                "Bucket": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2"
            }
        }
    ]
}
```

目标存储桶必须已启用版本控制。指定的角色必须具有写入目标存储桶的权限，并且必须建立允许 Amazon S3 代入角色的信任关系。

示例角色权限策略：

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetReplicationConfiguration",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET1/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete",
                "s3:ReplicateTags"
            ],
            "Resource": "arn:aws:s3:::AWSDOC-EXAMPLE-BUCKET2/*"
        }
    ]
}
```

示例信任关系策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Simple Storage Service 控制台用户指南》中的这是主题标题：

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketReplication。

PowerShell

适用于 PowerShell 的工具

示例 1：此示例使用单个规则设置复制配置，支持将存储桶“examplebucket”中使用键名称前缀“TaxDocs”创建的任何新对象复制到“exampletargetbucket”存储桶。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

示例 2：此示例设置具有多个规则的复制配置，支持将使用键名称前缀“TaxDocs”或“OtherDocs”创建的任何新对象复制到“exampletargetbucket”存储桶。键前缀不得重叠。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
```

```
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

示例 3：此示例更新指定存储桶上的复制配置，以禁用用于控制将键名称前缀为“TaxDocs”的对象复制到存储桶“exampletargetbucket”的规则。

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::exampletargetbucket" }

$params = @{
    BucketName = "examplebucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutBucketReplication。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketRequestPayment** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketRequestPayment。

CLI

AWS CLI

示例 1：为存储桶启用“申请方付款”配置

以下 put-bucket-request-payment 示例为指定的存储桶启用 requester pays。

```
aws s3api put-bucket-request-payment \
    --bucket my-bucket \
    --request-payment-configuration '{"Payer":"Requester"}'
```

此命令不生成任何输出。

示例 2：为存储桶禁用“申请方付款”配置

以下 put-bucket-request-payment 示例为指定的存储桶禁用 requester pays。

```
aws s3api put-bucket-request-payment \
    --bucket my-bucket \
    --request-payment-configuration '{"Payer":"BucketOwner"}'
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketRequestPayment。

PowerShell

适用于 PowerShell 的工具

示例 1：更新名为“mybucket”的存储桶的请求付款配置，以便向从该存储桶请求下载的人员收取下载费用。默认情况下，存储桶拥有者支付下载的费用。要将请求付款设置回默认值，请将“BucketOwner”用于 RequestPaymentConfiguration_Payer 参数。

```
Write-S3BucketRequestPayment -BucketName mybucket -
RequestPaymentConfiguration_Payer Requester
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutBucketRequestPayment。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketTagging** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketTagging。

CLI

AWS CLI

以下命令将标记配置应用于名为 my-bucket 的存储桶：

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging file://tagging.json
```

文件 tagging.json 是当前文件夹中指定标签的 JSON 文档：

```
{
   "TagSet": [
     {
       "Key": "organization",
       "Value": "marketing"
     }
   ]
}
```

或者，直接从命令行将标记配置应用于 my-bucket：

```
aws s3api put-bucket-tagging --bucket my-bucket --tagging
 'TagSet=[{Key=organization,Value=marketing}]'
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketTagging。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令将两个标签应用于名为 **cloudtrail-test-2018** 的存储桶：一个标签的键为 Stage，值为 Test；另一个标签的键为 Environment，值为 Alpha。要验证标签已添加到存储桶，请运行 **Get-S3BucketTagging -BucketName bucket_name**。结果应显示您在第一个命令中应用于存储桶的标签。请注意，**Write-S3BucketTagging** 会覆盖在存储桶上的整个现有标签集。要添加或删除各标签，请运行资源组和标记 API cmdlet **Add-RGTResourceTag** 和 **Remove-RGTResourceTag**。或者，使用 AWS 管理控制台中的标签编辑器来管理 S3 存储桶标签。

```
Write-S3BucketTagging -BucketName cloudtrail-test-2018 -TagSet @( @{ Key="Stage";
 Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

示例 2：此命令将名为 **cloudtrail-test-2018** 的存储桶传送到 **Write-S3BucketTagging** cmdlet。它将标签 Stage:Production 和 Department:Finance 应用于存储桶。请注意，**Write-S3BucketTagging** 会覆盖在存储桶上的整个现有标签集。

```
Get-S3Bucket -BucketName cloudtrail-test-2018 | Write-S3BucketTagging
 -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
 Value="Finance" } )
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutBucketTagging。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketVersioning** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketVersioning。

CLI

AWS CLI

以下命令对于名为 my-bucket 的存储桶启用版本控制。

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-
configuration Status=Enabled
```

以下命令启用版本控制，并使用 mfa 代码

```
aws s3api put-bucket-versioning --bucket my-bucket --versioning-
configuration Status=Enabled --mfa "SERIAL 123456"
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketVersioning。

PowerShell

适用于 PowerShell 的工具

示例 1：该命令为给定 S3 存储桶启用版本控制。

```
Write-S3BucketVersioning -BucketName 's3testbucket' -VersioningConfig_Status
 Enabled
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutBucketVersioning。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutBucketWebsite** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutBucketWebsite。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
                // Put the website configuration.
                PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument,
                    },
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- 有关更多信息，请参阅《AWS SDK for .NET API 参考》中的 PutBucketWebsite。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
&errorPage,
                                  const Aws::S3::S3ClientConfiguration
&clientConfig) {
   Aws::S3::S3Client client(clientConfig);

   Aws::S3::Model::IndexDocument indexDocument;
   indexDocument.SetSuffix(indexPage);

   Aws::S3::Model::ErrorDocument errorDocument;
   errorDocument.SetKey(errorPage);
```

```
    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
            client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关更多信息，请参阅《AWS SDK for C++ API 参考》中的 PutBucketWebsite。

CLI

AWS CLI

将静态网站配置应用于名为 `my-bucket` 的存储桶：

```
aws s3api put-bucket-website --bucket my-bucket --website-configuration file://
website.json
```

文件 `website.json` 是当前文件夹中的一个 JSON 文档，用于指定网站的索引和错误页面：

```
{
    "IndexDocument": {
        "Suffix": "index.html"
    },
    "ErrorDocument": {
        "Key": "error.html"
```

```
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutBucketWebsite。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucketName> [indexdoc]\s

                Where:
                   bucketName   - The Amazon S3 bucket to set the website
 configuration on.\s
                   indexdoc - The index document, ex. 'index.html'
```

```
                                If not specified, 'index.html' will be set.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    public static void setWebsiteConfig(S3Client s3, String bucketName, String
 indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()

 .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                    .build();

            PutBucketWebsiteRequest pubWebsiteReq =
 PutBucketWebsiteRequest.builder()
                    .bucket(bucketName)
                    .websiteConfiguration(websiteConfig)
                    .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关更多信息，请参阅《AWS SDK for Java 2.x API 参考》中的 PutBucketWebsite。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

设置网站配置。

```
import { PutBucketWebsiteCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

// Set up a bucket as a static website.
// The bucket needs to be publicly accessible.
export const main = async () => {
  const command = new PutBucketWebsiteCommand({
    Bucket: "test-bucket",
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request that is for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关更多信息，请参阅《AWS SDK for JavaScript API 参考》中的 PutBucketWebsite。

PowerShell

适用于 PowerShell 的工具

示例 1：该命令为给定存储桶启用网站托管，索引文档为“index.html”，错误文档为“error.html”。

```
Write-S3BucketWebsite -BucketName 's3testbucket' -
WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutBucketWebsite。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end
```

```
  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
 website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关更多信息，请参阅《AWS SDK for Ruby API 参考》中的 PutBucketWebsite。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObject。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基础知识
- 跟踪上传和下载操作
- 使用 Amazon S3 对象完整性

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public static async Task<bool> UploadFileAsync(
    IAmazonS3 client,
    string bucketName,
    string objectName,
    string filePath)
{
```

```
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await client.PutObjectAsync(request);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully uploaded {objectName} to
{bucketName}.");
                return true;
            }
            else
            {
                Console.WriteLine($"Could not upload {objectName} to
{bucketName}.");
                return false;
            }
        }
```

使用服务器端加密上传对象。

```
    using System;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket with server-side encryption enabled.
    /// </summary>
    public class ServerSideEncryption
    {
        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "samplefile.txt";

            // If the AWS Region defined for your default user is different
```

```
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            await WritingAnObjectAsync(client, bucketName, keyName);
        }

        /// <summary>
        /// Upload a sample object include a setting for encryption.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// to upload a file and apply server-side encryption.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
        /// encrypted object will reside.</param>
        /// <param name="keyName">The name for the object that you want to
        /// create in the supplied bucket.</param>
        public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
                };

                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
                GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;
```

```
                Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
            }
        }
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for .NET API 参考中的 PutObject。

Bash

AWS CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#       $1 - The name of the bucket to copy the file to.
```

```
#       $2 - The path and file name of the local file to copy to the bucket.
#       $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function copy_file_to_bucket() {
  local response bucket_name source_file destination_file_name
  bucket_name=$1
  source_file=$2
  destination_file_name=$3

  response=$(aws s3api put-object \
    --bucket "$bucket_name" \
    --body "$source_file" \
    --key "$destination_file_name")

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
  fi
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutObject。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);
```

```
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          fileName.c_str(),
                                          std::ios_base::in |
std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Added object '" << fileName << "' to bucket '"
                  << bucketName << "'.";
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅 AWS SDK for C++ API 参考中的 PutObject。

CLI

AWS CLI

以下示例使用 `put-object` 命令将对象上传到 Amazon S3：

```
aws s3api put-object --bucket text-content --key dir-1/my_images.tar.bz2 --
body my_images.tar.bz2
```

以下示例演示了如何上传视频文件（该视频文件是使用 Windows 文件系统语法指定的。）：

```
aws s3api put-object --bucket text-content --key dir-1/big-video-file.mp4 --body
 e:\media\videos\f-sharp-3-data-services.mp4
```

有关上传对象的更多信息，请参阅《Amazon S3 开发人员指南》中的“上传对象”。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutObject。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用低级别 API 将对象放入存储桶。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(bucketName string, objectKey string,
 fileName string) error {
 file, err := os.Open(fileName)
```

```
 if err != nil {
  log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
 } else {
  defer file.Close()
  _, err = basics.S3Client.PutObject(context.TODO(), &s3.PutObjectInput{
   Bucket: aws.String(bucketName),
   Key:    aws.String(objectKey),
   Body:   file,
  })
  if err != nil {
   log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
    fileName, bucketName, objectKey, err)
  }
 }
 return err
}
```

使用传输管理器将对象上传到存储桶。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
```

```
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}
```

- 有关 API 详细信息，请参阅 AWS SDK for Go API 参考中的 PutObject。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用 S3Client 将文件上传到存储桶。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
```

```
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <bucketName> <objectKey> <objectPath>\s

                Where:
                  bucketName - The Amazon S3 bucket to upload an object into.
                  objectKey - The object to upload (for example, book.pdf).
                  objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }
```

```
    // This example uses RequestBody.fromFile to avoid loading the whole file
 into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

使用 S3TransferManager 将文件上传到存储桶。查看完整文件并进行测试。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
 bucketName,
```

```
                            String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

使用 S3Client 将对象上传到存储桶并设置标签。

```
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()
                    .key("Tag 1")
                    .value("This is tag 1")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Tag 2")
                    .value("This is tag 2")
                    .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag1);
            tags.add(tag2);

            Tagging allTags = Tagging.builder()
                    .tagSet(tags)
                    .build();

            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .tagging(allTags)
                    .build();
```

```
            s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
        try {
            GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
            List<Tag> obTags = getTaggingRes.tagSet();
            for (Tag sinTag : obTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

            // Replace the object's tags with two new tags.
            Tag tag3 = Tag.builder()
                    .key("Tag 3")
                    .value("This is tag 3")
                    .build();

            Tag tag4 = Tag.builder()
                    .key("Tag 4")
                    .value("This is tag 4")
                    .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag3);
            tags.add(tag4);

            Tagging updatedTags = Tagging.builder()
                    .tagSet(tags)
                    .build();
```

```
            PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .tagging(updatedTags)
                    .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Return a byte array.
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
```

```
        }

        return bytesArray;
    }
}
```

使用 S3Client 将对象上传到存储桶并设置元数据。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

                Usage:
                  <bucketName> <objectKey> <objectPath>\s

                Where:
                  bucketName - The Amazon S3 bucket to upload an object into.
                  objectKey - The object to upload (for example, book.pdf).
                  objectPath - The path where the file is located (for example,
 C:/AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(USAGE);
```

```
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file
 into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String
 objectKey, String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .metadata(metadata)
                    .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket
 " + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

使用 S3Client 将对象上传到存储桶并设置对象保留值。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <key> <bucketName>\s

                Where:
                    key - The name of the object (for example, book.pdf).\s
                    bucketName - The Amazon S3 bucket name that contains the
 object (for example, bucket1).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
```

```
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                    .mode("COMPLIANCE")
                    .retainUntilDate(instant)
                    .build();

            PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
                    .bucket(bucket)
                    .key(key)
                    .bypassGovernanceRetention(true)
                    .retention(lockRetention)
                    .build();

            // To set Retention on an object, the Amazon S3 bucket must support
 object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
 placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 PutObject。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

上传对象。

```
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";

const client = new S3Client({});

export const main = async () => {
  const command = new PutObjectCommand({
    Bucket: "test-bucket",
    Key: "hello-s3.txt",
    Body: "Hello S3!",
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。
- 有关 API 详细信息，请参阅 AWS SDK for JavaScript API 参考中的 PutObject。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 PutObject。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

将对象上传到存储桶。

```
        $s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

        $fileName = __DIR__ . "/local-file-" . uniqid();
        try {
            $this->s3client->putObject([
                'Bucket' => $this->bucketName,
                'Key' => $fileName,
                'SourceFile' => __DIR__ . '/testfile.txt'
            ]);
            echo "Uploaded $fileName to $this->bucketName.\n";
        } catch (Exception $exception) {
            echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
            exit("Please fix error with file upload before continuing.");
        }
```

- 有关 API 的详细信息，请参阅 AWS SDK for PHP API 参考中的 PutObject。

PowerShell

适用于 PowerShell 的工具

示例 1：此命令将单个文件“local-sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“sample.txt”的对象。

```
Write-S3Object -BucketName test-files -Key "sample.txt" -File .\local-sample.txt
```

示例 2：此命令将单个文件“sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“sample.txt”的对象。如果未提供 -Key 参数，则文件名将用作 S3 对象键。

```
Write-S3Object -BucketName test-files -File .\sample.txt
```

示例 3：此命令将单个文件“local-sample.txt”上传到 Amazon S3，同时在存储桶“test-files”中创建键为“prefix/to/sample.txt”的对象。

```
Write-S3Object -BucketName test-files -Key "prefix/to/sample.txt" -File .\local-
sample.txt
```

示例 4：此命令将子目录“Scripts”中的所有文件上传到存储桶“test-files”，并将公用键前缀“SampleScripts”应用于每个对象。每个上传的文件都将有一个键“SampleScripts/filename”，其中“filename”不同。

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
```

示例 5：此命令将本地目录“Scripts”中的所有 *.ps1 文件上传到存储桶“test-files”，并将公用键前缀“SampleScripts”应用于每个对象。每个上传的文件都将有一个键“SampleScripts/filename.ps1”，其中“filename”不同。

```
Write-S3Object -BucketName test-files -Folder .\Scripts -KeyPrefix SampleScripts\
 -SearchPattern *.ps1
```

示例 6：此命令创建一个新的 S3 对象，其中包含键为“sample.txt”的指定内容字符串。

```
Write-S3Object -BucketName test-files -Key "sample.txt" -Content "object
 contents"
```

示例 7：此命令上传指定的文件（文件名用作键），并将指定的标签应用于新对象。

```
Write-S3Object -BucketName test-files -File "sample.txt" -TagSet
 @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

示例 8：此命令以递归方式上传指定的文件夹，并将指定的标签应用于所有新对象。

```
Write-S3Object -BucketName test-files -Folder . -KeyPrefix "TaggedFiles" -Recurse
 -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutObject。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
 in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
 When this
                     argument is a string, it is interpreted as a file name,
 which is
                     opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.",
 data)
```

```
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):
                put_data.close()
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 PutObject。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用托管上传工具 (Object.upload_file) 上传文件。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 Object.put 上传文件。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

使用 Object.put 上传文件并添加服务器端加密。

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
```

```
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅 AWS SDK for Ruby API 参考中的 PutObject。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<PutObjectOutput, SdkError<PutObjectError>> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 PutObject。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    "Get contents of file from application server."
    DATA lv_body TYPE xstring.
    OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
    READ DATASET iv_file_name INTO lv_body.
    CLOSE DATASET iv_file_name.

    "Upload/put an object to an S3 bucket."
    TRY.
        lo_s3->putobject(
            iv_bucket = iv_bucket_name
            iv_key = iv_file_name
```

```
            iv_body = lv_body
        ).
        MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
      CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《AWS SDK for SAP ABAP API 参考》中的 PutObject。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

将文件从本地存储上传到存储桶。

```
    public func uploadFile(bucket: String, key: String, file: String) async
throws {
        let fileUrl = URL(fileURLWithPath: file)
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )
        _ = try await client.putObject(input: input)
    }
```

将 Swift 数据对象的内容上传到存储桶。

```
   public func createFile(bucket: String, key: String, withData data: Data)
async throws {
       let dataStream = ByteStream.data(data)

       let input = PutObjectInput(
           body: dataStream,
           bucket: bucket,
           key: key
       )
       _ = try await client.putObject(input: input)
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 PutObject。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObjectAcl** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObjectAcl。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理访问控制列表（ACL）

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
 &objectKey, const Aws::String &ownerID,
                               const Aws::String &granteePermission, const
 Aws::String &granteeType,
                               const Aws::String &granteeID, const Aws::String
 &granteeEmailAddress,
                               const Aws::String &granteeURI, const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);
```

```
    Aws::S3::Model::PutObjectAclOutcome outcome =
            s3Client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
                  << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param access: Human readable string.
 \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
 enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
```

```
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 PutObjectAcl。

CLI

AWS CLI

以下示例向两个 AWS 用户（user1@example.com 和user2@example.com）授予 `full control`，并向所有人授予 `read`：

```
aws s3api put-object-acl --bucket MyBucket --key file.txt --grant-full-
control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-
read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

有关自定义 ACL（s3api ACL 命令（如 `put-object-acl`）使用相同的速记参数表示法）的详细信息，请参阅 http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutObjectAcl。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class ObjectWrapper:
```

```
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key


    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user
identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                    "Permission": "READ",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 PutObjectAcl。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObjectLegalHold** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObjectLegalHold。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 锁定 Amazon S3 对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="holdStatus">The On or Off status for the legal hold.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
            {
                Status = holdStatus
```

```
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 PutObjectLegalHold。

CLI

AWS CLI

对对象应用法定保留

以下 put-object-legal-hold 示例在 doc1.rtf 对象上设置了法定保留。

```
aws s3api put-object-legal-hold \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf \
    --legal-hold Status=ON
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutObjectLegalHold。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }
```

```
 return err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 PutObjectLegalHold。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   // Set or modify a legal hold on an object in an S3 bucket.
   public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
       ObjectLockLegalHold legalHold ;
       if (legalHoldOn) {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.ON)
               .build();
       } else {
           legalHold = ObjectLockLegalHold.builder()
               .status(ObjectLockLegalHoldStatus.OFF)
               .build();
       }

       PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
           .bucket(bucketName)
           .key(objectKey)
           .legalHold(legalHold)
           .build();

       getClient().putObjectLegalHold(legalHoldRequest) ;
       System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
```

```
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 PutObjectLegalHold。

JavaScript

适用于 JavaScript 的 SDK（v3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import { fileURLToPath } from "url";
import { PutObjectLegalHoldCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: "ON",
    },
    // Optionally, you can provide additional parameters
    // ChecksumAlgorithm: "ALGORITHM",
    // ContentMD5: "MD5_HASH",
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
```

```
    console.log(
      `Object legal hold status: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 PutObjectLegalHold。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

放置对象法定保留。

```
def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to set the legal hold on.
    """
    print()
    logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
    try:
```

```
        before_status = "OFF"
        after_status = "ON"
        s3_client.put_object_legal_hold(
            Bucket=bucket, Key=key, LegalHold={"Status": after_status}
        )
        logger.debug(
            "Legal hold set successfully on file [%s] in bucket [%s]", key,
bucket
        )
        _print_legal_hold_update(bucket, key, before_status, after_status)
    except Exception as e:
        logger.error(
            "Failed to set legal hold on file [%s] in bucket [%s]: %s", key,
bucket, e
        )
```

- 有关 API 详细信息，请参阅《适用于 Python 的 AWS SDK（Boto3）API 参考》中的 PutObjectLegalHold。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObjectLockConfiguration** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObjectLockConfiguration。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 锁定 Amazon S3 对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

设置存储桶的对象锁定配置。

```
    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };
```

```
            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

设置存储桶的默认保留期。

```
    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
```

```
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 PutObjectLockConfiguration。

CLI

AWS CLI

在存储桶上设置对象锁定配置

以下 put-object-lock-configuration 示例在指定存储桶上设置了 50 天的对象锁定。

```
aws s3api put-object-lock-configuration \
```

```
    --bucket my-bucket-with-object-lock \
    --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
{ "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutObjectLockConfiguration。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

设置存储桶的对象锁定配置。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
```

```
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

设置存储桶的默认保留期。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
```

```
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 PutObjectLockConfiguration。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

设置存储桶的对象锁定配置。

```
   // Enable object lock on an existing bucket.
   public void enableObjectLockOnBucket(String bucketName) {
       try {
           VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
               .status(BucketVersioningStatus.ENABLED)
               .build();

           PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
               .bucket(bucketName)
               .versioningConfiguration(versioningConfiguration)
               .build();

           // Enable versioning on the bucket.
           getClient().putBucketVersioning(putBucketVersioningRequest);
           PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
               .bucket(bucketName)
               .objectLockConfiguration(ObjectLockConfiguration.builder()
                   .objectLockEnabled(ObjectLockEnabled.ENABLED)
                   .build())
               .build();

           getClient().putObjectLockConfiguration(request);
           System.out.println("Successfully enabled object lock on
"+bucketName);

       } catch (S3Exception ex) {
           System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
       }
   }
```

设置存储桶的默认保留期。

```
   // Set or modify a retention period on an S3 bucket.
   public void modifyBucketDefaultRetention(String bucketName) {
       VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
```

```
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
+".");
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 PutObjectLockConfiguration。

JavaScript

适用于 JavaScript 的 SDK（v3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

设置存储桶的对象锁定配置。

```
import { fileURLToPath } from "url";
import {
  PutObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // Token: "OPTIONAL_TOKEN",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object Lock Configuration updated: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
```

```
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

设置存储桶的默认保留期。

```
import { fileURLToPath } from "url";
import {
  PutObjectLockConfigurationCommand,
  S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 */
export const main = async (client, bucketName) => {
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
 bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
      Rule: {
        DefaultRetention: {
          Mode: "GOVERNANCE",
          Years: 3,
        },
      },
    },
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // Token: "OPTIONAL_TOKEN",
  });

  try {
    const response = await client.send(command);
```

```
    console.log(
      `Default Object Lock Configuration updated: ${response.
$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME");
}
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 PutObjectLockConfiguration。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

放置对象锁定配置。

```
        s3_client.put_object_lock_configuration(
            Bucket=bucket,
            ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":
{}},
        )
```

- 有关 API 详细信息，请参阅《适用于 Python 的 AWS SDK（Boto3）API 参考》中的 PutObjectLockConfiguration。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutObjectRetention** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutObjectRetention。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 锁定 Amazon S3 对象

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
```

```
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
 until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
 '{ex.Message}'");
            return false;
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 PutObjectRetention。

CLI

AWS CLI

为对象设置对象保留配置

以下 put-object-retention 示例为指定对象设置直到 2025 年 1 月 1 日的对象保留配置。

```
aws s3api put-object-retention \
    --bucket my-bucket-with-object-lock \
    --key doc1.rtf \
    --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":
 "2025-01-01T00:00:00" }'
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 PutObjectRetention。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }
```

```
    return err
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的 PutObjectRetention。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   // Set or modify a retention period on an object in an S3 bucket.
   public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
       // Calculate the instant one day from now.
       Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

       // Convert the Instant to a ZonedDateTime object with a specific time
zone.
       ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

       // Define a formatter for human-readable output.
       DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

       // Format the ZonedDateTime object to a human-readable date string.
       String humanReadableDate = formatter.format(zonedDateTime);

       // Print the formatted date string.
       System.out.println("Formatted Date: " + humanReadableDate);
       ObjectLockRetention retention = ObjectLockRetention.builder()
           .mode(ObjectLockRetentionMode.GOVERNANCE)
           .retainUntilDate(futureInstant)
           .build();
```

```
        PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
 until "+ humanReadableDate +".");
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 PutObjectRetention。

JavaScript

适用于 JavaScript 的 SDK (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import { fileURLToPath } from "url";
import { PutObjectRetentionCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: objectKey,
    BypassGovernanceRetention: false,
    // ChecksumAlgorithm: "ALGORITHM",
    // ContentMD5: "MD5_HASH",
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
```

```
    Retention: {
      Mode: "GOVERNANCE", // or "COMPLIANCE"
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(
      `Object Retention settings updated: ${response.$metadata.httpStatusCode}`,
    );
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "BUCKET_NAME", "OBJECT_KEY");
}
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 PutObjectRetention。

PowerShell

适用于 PowerShell 的工具

示例 1：该命令为给定 S3 存储桶中的“testfile.txt”对象启用监管保留模式，直到日期“2019 年 12 月 31 日 00:00:00”。

```
Write-S3ObjectRetention -BucketName 's3buckettesting' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 PutObjectRetention。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

放置对象保留。

```
            s3_client.put_object_retention(
                Bucket=bucket,
                Key=key,
                VersionId=version_id,
                Retention={"Mode": "GOVERNANCE", "RetainUntilDate":
far_future_date},
                BypassGovernanceRetention=True,
            )
```

- 有关 API 详细信息，请参阅《适用于 Python 的 AWS SDK（Boto3）API 参考》中的 PutObjectRetention。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **RestoreObject** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `RestoreObject`。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   using System;
   using System.Threading.Tasks;
   using Amazon;
   using Amazon.S3;
   using Amazon.S3.Model;

   /// <summary>
   /// This example shows how to restore an archived object in an Amazon
   /// Simple Storage Service (Amazon S3) bucket.
   /// </summary>
   public class RestoreArchivedObject
   {
       public static void Main()
       {
           string bucketName = "doc-example-bucket";
           string objectKey = "archived-object.txt";

           // Specify your bucket region (an example region is shown).
           RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

           IAmazonS3 client = new AmazonS3Client(bucketRegion);
           RestoreObjectAsync(client, bucketName, objectKey).Wait();
       }

       /// <summary>
       /// This method restores an archived object from an Amazon S3 bucket.
       /// </summary>
       /// <param name="client">The initialized Amazon S3 client object used to
call
       /// RestoreObjectAsync.</param>
       /// <param name="bucketName">A string representing the name of the
       /// bucket where the object was located before it was archived.</param>
       /// <param name="objectKey">A string representing the name of the
       /// archived object to restore.</param>
       public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
       {
           try
           {
               var restoreRequest = new RestoreObjectRequest
               {
                   BucketName = bucketName,
                   Key = objectKey,
```

```
                    Days = 2,
                };
                RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

                // Check the status of the restoration.
                await CheckRestorationStatusAsync(client, bucketName, objectKey);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Error: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// This method retrieves the status of the object's restoration.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// GetObjectMetadataAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
        /// S3 bucket which contains the archived object.</param>
        /// <param name="objectKey">A string representing the name of the
        /// archived object you want to restore.</param>
        public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
        {
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
            };

            GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

            var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
            Console.WriteLine($"Restoration status: {restStatus}");
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 RestoreObject。

CLI

AWS CLI

为对象创建还原请求

以下 `restore-object` 示例将存储桶 `my-glacier-bucket` 的指定 Amazon S3 Glacier 对象还原为 10 天。

```
aws s3api restore-object \
    --bucket my-glacier-bucket \
    --key doc1.rtf \
    --restore-request Days=10
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 RestoreObject。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
```

```
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 *  Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName> <keyName> <expectedBucketOwner>

                Where:
                    bucketName - The Amazon S3 bucket name.\s
                    keyName - The key name of an object with a Storage class
 value of Glacier.\s
                    expectedBucketOwner - The account that owns the bucket (you
 can obtain this value from the AWS Management Console).\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }
```

```
    public static void restoreS3Object(S3Client s3, String bucketName, String
 keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                    .days(10)

 .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                    .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                    .expectedBucketOwner(expectedBucketOwner)
                    .bucket(bucketName)
                    .key(keyName)
                    .restoreRequest(restoreRequest)
                    .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 RestoreObject。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **SelectObjectContent** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 SelectObjectContent。

CLI

AWS CLI

基于 SQL 语句筛选 Amazon S3 对象的内容

以下 select-object-content 示例使用指定的 SQL 语句筛选 my-data-file.csv 对象并将输出发送到文件。

```
aws s3api select-object-content \
    --bucket my-bucket \
    --key my-data-file.csv \
    --expression "select * from s3object limit 100" \
    --expression-type 'SQL' \
    --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
    --output-serialization '{"CSV": {}}' "output.csv"
```

此命令不生成任何输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 SelectObjectContent。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

以下示例演示了一个使用 JSON 对象的查询。该完整示例还演示了 CSV 对象的用法。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
```

```
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
 software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
 LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" +
 UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
```

```
        final String queryExpression = "select * from s3object[*][*] c where
c.area < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
                .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
handler.
        // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
        s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
```

```
    }

    static SelectObjectContentResponseHandler
 buildResponseHandler(EventStreamInfo eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
 information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
 SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
                    eventStreamInfo.incrementOnRecordsCalled();
                })
                .onCont(ce -> {
                    logger.info("Continuation event received.");
                    eventStreamInfo.incrementContinuationEvents();
                })
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
 processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
```

```
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 SelectObjectContent。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UploadPart** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 UploadPart。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 执行分段上传
- 使用校验和
- 使用 Amazon S3 对象完整性

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
 checksum algorithm,
                           ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
 &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
 &uploadID,
```

```
                                                        int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                        const Aws::String
 &calculatedHash,
                                                        const
 std::shared_ptr<Aws::IOStream> &body,
                                                        const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 UploadPart。

CLI

AWS CLI

以下命令上传使用 create-multipart-upload 命令启动的分段上传中的第一个分段：

```
aws s3api upload-part --bucket my-bucket --key 'multipart/01' --part-number 1 --
body part01 --upload-id
  "dfRtDYU0WWCCcH43C3WFbkRONycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

body 选项采用本地文件的名称或路径进行上传（不要使用 file:// 前缀）。最小分段大小为 5 MB。上传 ID 由 create-multipart-upload 返回，也可以使用 list-multipart-uploads 进行检索。存储桶和键是在您创建分段上传时指定的。

输出：

```
{
    "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

保存每个分段的 ETag 值以备后用。需要这些值才能完成分段上传。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 UploadPart。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
```

```
            .part_number(part_number)
            .send()
            .await?;
        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );

   let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 UploadPart。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

适用于使用 AWS 软件开发工具包的 Amazon S3 的场景

以下代码示例显示如何通过 AWS SDK 实施 Amazon S3 中的常见场景。这些场景向您展示了如何通过调用 Amazon S3 中的多个函数或与其它 AWS 服务结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 使用 AWS SDK 将文本转换为语音以及将语音转换回文本
- 使用 AWS SDK 创建 Amazon S3 的预签名 URL
- 创建照片资产管理应用程序，让用户能够使用标签管理照片
- 使用 AWS SDK 列出 Amazon S3 对象的网页
- 创建 Amazon Textract 浏览器应用程序
- 使用 AWS SDK 删除未完成的 Amazon S3 分段上传
- 使用 AWS SDK 通过 Amazon Rekognition 检测图像中的 PPE

- 使用 AWS SDK 检测从图像中提取的文本中的实体
- 使用 AWS SDK 检测图像中的人脸
- 使用 AWS 软件开发工具包通过 Amazon Rekognition 检测图像中的对象
- 使用 AWS SDK 通过 Amazon Rekognition 检测视频中的人物和对象
- 将 Amazon Simple Storage Service（Amazon S3）桶中的所有对象下载到本地目录
- 使用 AWS SDK 从多区域接入点中获取 Amazon S3 对象
- 使用 AWS SDK 并指定 If-Modified-Since 标头以从 Amazon S3 存储桶获取对象
- 使用 AWS SDK 加密 Amazon S3 对象入门
- 使用 AWS SDK 为 Amazon S3 存储桶和对象加标签入门
- 使用 AWS SDK 获取 Amazon S3 存储桶的法定保留配置
- 通过 AWS SDK 使用 Amazon S3 对象锁定功能
- 使用 AWS SDK 管理 Amazon S3 存储桶的访问控制列表（ACL）
- 通过 AWS SDK 使用 Lambda 函数批量管理版本控制的 Amazon S3 对象
- 使用 AWS SDK 解析 Amazon S3 URI
- 使用 AWS SDK 执行 Amazon S3 对象的分段复制
- 使用 AWS SDK 执行 Amazon S3 对象的分段上传
- 使用 AWS SDK 接收和处理 Amazon S3 事件通知。
- 使用 AWS SDK 保存 EXIF 和其他图像信息
- 使用 AWS SDK 向 Amazon EventBridge 发送 S3 事件通知
- 跟踪使用 AWS SDK 执行的 Amazon S3 对象上传或下载操作
- 使用 S3 对象 Lambda 转换应用程序的数据
- 使用 AWS SDK 进行单元测试和集成测试的示例方法
- 以递归方式将本地目录上传到 Amazon Simple Storage Service（Amazon S3）桶
- 借助 AWS SDK 向 Amazon S3 上传大文件或从 Amazon S3 下载大文件
- 使用 AWS SDK 将未知大小的流上传到 Amazon S3 对象
- 使用 AWS 开发工具包，对 Amazon S3 对象使用校验和
- 通过 AWS SDK 使用 Amazon S3 对象完整性功能
- 使用 AWS SDK 使用 Amazon S3 版本控制对象

使用 AWS SDK 将文本转换为语音以及将语音转换回文本

以下代码示例展示了如何：

- 使用 Amazon Polly 将纯文本 (UTF-8) 输入文件合成为音频文件。
- 将音频文件上传到 Amazon S3 存储桶。
- 使用 Amazon Transcribe 将音频文件转换为文本。
- 显示文本。

Rust

适用于 Rust 的 SDK

使用 Amazon Polly 将纯文本（UTF-8）输入文件合成为音频文件，将音频文件上传到 Amazon S3 存储桶，使用 Amazon Transcribe 将该音频文件转换为文本，然后显示文本。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Polly
- Amazon S3
- Amazon Transcribe

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 创建 Amazon S3 的预签名 URL

以下代码示例显示如何为 Amazon S3 创建预签名 URL 以及如何上传对象。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

生成可在有限时间内执行 Amazon S3 操作的预签名 URL。

```
    using System;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;

    public class GenPresignedUrl
    {
        public static void Main()
        {
            const string bucketName = "doc-example-bucket";
            const string objectKey = "sample.txt";

            // Specify how long the presigned URL lasts, in hours
            const double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

            string urlString = GeneratePresignedURL(s3Client, bucketName,
 objectKey, timeoutDuration);
            Console.WriteLine($"The generated URL is: {urlString}.");
        }

        /// <summary>
        /// Generate a presigned URL that can be used to access the file named
        /// in the objectKey parameter for the amount of time specified in the
        /// duration parameter.
        /// </summary>
        /// <param name="client">An initialized S3 client object used to call
```

```
        /// the GetPresignedUrl method.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// object for which to create the presigned URL.</param>
        /// <param name="objectKey">The name of the object to access with the
        /// presigned URL.</param>
        /// <param name="duration">The length of time for which the presigned
        /// URL will be valid.</param>
        /// <returns>A string representing the generated presigned URL.</returns>
        public static string GeneratePresignedURL(IAmazonS3 client, string
bucketName, string objectKey, double duration)
        {
            string urlString = string.Empty;
            try
            {
                var request = new GetPreSignedUrlRequest()
                {
                    BucketName = bucketName,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration),
                };
                urlString = client.GetPreSignedURL(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:'{ex.Message}'");
            }

            return urlString;
        }
    }
```

生成预签名 URL 并使用该 URL 执行上传。

```
    using System;
    using System.IO;
    using System.Net.Http;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.S3;
    using Amazon.S3.Model;
```

```
    /// <summary>
    /// This example shows how to upload an object to an Amazon Simple Storage
    /// Service (Amazon S3) bucket using a presigned URL. The code first
    /// creates a presigned URL and then uses it to upload an object to an
    /// Amazon S3 bucket using that URL.
    /// </summary>
    public class UploadUsingPresignedURL
    {
        private static HttpClient httpClient = new HttpClient();

        public static async Task Main()
        {
            string bucketName = "doc-example-bucket";
            string keyName = "samplefile.txt";
            string filePath = $"source\\{keyName}";

            // Specify how long the signed URL will be valid in hours.
            double timeoutDuration = 12;

            // Specify the AWS Region of your Amazon S3 bucket. If it is
            // different from the Region defined for the default user,
            // pass the Region to the constructor for the client. For
            // example: new AmazonS3Client(RegionEndpoint.USEast1);

            // If using the Region us-east-1, and server-side encryption with AWS
 KMS, you must specify Signature Version 4.
            // Region us-east-1 defaults to Signature Version 2 unless explicitly
 set to Version 4 as shown below.
            // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/
userguide/UsingAWSSDK.html#specify-signature-version
            // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/
TAWSConfigsS3.html
            AWSConfigsS3.UseSignatureVersion4 = true;
            IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

            var url = GeneratePreSignedURL(client, bucketName, keyName,
 timeoutDuration);
            var success = await UploadObject(filePath, url);

            if (success)
            {
                Console.WriteLine("Upload succeeded.");
            }
            else
```

```
            {
                Console.WriteLine("Upload failed.");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket using the presigned URL
 passed in
        /// the url parameter.
        /// </summary>
        /// <param name="filePath">The path (including file name) to the local
        /// file you want to upload.</param>
        /// <param name="url">The presigned URL that will be used to upload the
        /// file to the Amazon S3 bucket.</param>
        /// <returns>A Boolean value indicating the success or failure of the
        /// operation, based on the HttpWebResponse.</returns>
        public static async Task<bool> UploadObject(string filePath, string url)
        {
            using var streamContent = new StreamContent(
                new FileStream(filePath, FileMode.Open, FileAccess.Read));

            var response = await httpClient.PutAsync(url, streamContent);
            return response.IsSuccessStatusCode;
        }

        /// <summary>
        /// Generates a presigned URL which will be used to upload an object to
        /// an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetPreSignedURL.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
 the
        /// presigned URL will point.</param>
        /// <param name="objectKey">The name of the file that will be uploaded.</
param>
        /// <param name="duration">How long (in hours) the presigned URL will
        /// be valid.</param>
        /// <returns>The generated URL.</returns>
        public static string GeneratePreSignedURL(
            IAmazonS3 client,
            string bucketName,
            string objectKey,
```

```
            double duration)
        {
            var request = new GetPreSignedUrlRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Verb = HttpVerb.PUT,
                Expires = DateTime.UtcNow.AddHours(duration),
            };

            string url = client.GetPreSignedURL(request);
            return url;
        }
    }
```

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

生成预签名 URL 来下载对象。

```
//! Routine which demonstrates creating a pre-signed URL to download an object
 from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param expirationSeconds: Expiration in seconds for pre-signed URL.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
 &bucketName,
                                                      const Aws::String &key,
```

```
                                                        uint64_t expirationSeconds,
                                                        const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_GET,
                                       expirationSeconds);
}
```

使用 libcurl 下载。

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to get an object from a bucket.
  \param resultString: A string to hold the result.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                 Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);
```

```
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
 std::endl;
        return false;
    }

    return true;
}
```

生成预签名 URL 来上传对象。

```
//! Routine which demonstrates creating a pre-signed URL to upload an object to
 an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
  \param bucketName: Name of the bucket.
  \param key: Name of an object key.
  \param clientConfig: Aws client configuration.
  \return Aws::String: A pre-signed URL.
*/
```

```
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
 &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,
                                                       const
 Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
 Aws::Http::HttpMethod::HTTP_PUT,
                                       expirationSeconds);
}
```

使用 libcurl 上传。

```
static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
 *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to put an object in a bucket.
  \param data: Body of the putObject request.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                 const Aws::String &data) {
    CURL *curl = curl_easy_init();
```

```
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
```

```
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    std::string outString = outWriteString.str();
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
                  << std::endl;
        return false;
    }
}
```

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建包装 S3 预签名操作的函数。

```
// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
 actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
 S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
 client.
type Presigner struct {
 PresignClient *s3.PresignClient
}



// GetObject makes a presigned request that can be used to get an object from a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
 bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignGetObject(context.TODO(),
 &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// PutObject makes a presigned request that can be used to put an object in a
 bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
 bucketName string, objectKey string, lifetimeSecs int64)
 (*v4.PresignedHTTPRequest, error) {
```

```
 request, err := presigner.PresignClient.PresignPutObject(context.TODO(),
 &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 }, func(opts *s3.PresignOptions) {
  opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return request, err
}



// DeleteObject makes a presigned request that can be used to delete an object
 from a bucket.
func (presigner Presigner) DeleteObject(bucketName string, objectKey string)
 (*v4.PresignedHTTPRequest, error) {
 request, err := presigner.PresignClient.PresignDeleteObject(context.TODO(),
 &s3.DeleteObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
 %v\n", objectKey, err)
 }
 return request, err
}
```

运行一个交互式示例，以生成并使用预签名 URL 上传、下载和删除 S3 对象。

```

// RunPresigningScenario is an interactive example that shows you how to get
 presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
 Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
 can
```

```
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local
 file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
 object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
 so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
 mocked
// during testing.
func RunPresigningScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner, httpRequester IHttpRequester) {
 defer func() {
  if r := recover(); r != nil {
   fmt.Printf("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 presigning demo.")
 log.Println(strings.Repeat("-", 88))

 s3Client := s3.NewFromConfig(sdkConfig)
 bucketBasics := actions.BucketBasics{S3Client: s3Client}
 presignClient := s3.NewPresignClient(s3Client)
 presigner := actions.Presigner{PresignClient: presignClient}

 bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
  "you own or one you want to create:", demotools.NotEmpty{})
 bucketExists, err := bucketBasics.BucketExists(bucketName)
 if err != nil {
  panic(err)
```

```
}
if !bucketExists {
 err = bucketBasics.CreateBucket(bucketName, sdkConfig.Region)
 if err != nil {
  panic(err)
 } else {
  log.Println("Bucket created.")
 }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
 demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
 demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
 panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
 presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
 panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
 uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
```

```
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(bucketName, uploadKey, 60)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
 presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
 uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
 panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(bucketName, uploadKey)
if err != nil {
 panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
 presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
 panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.\n",
presignedDelRequest.Method,
 uploadKey, delResponse.StatusCode)
```

```
 log.Println(strings.Repeat("-", 88))

 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

定义示例用于发出 HTTP 请求的 HTTP 请求包装器。

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
 during
// unit testing.
type IHttpRequester interface {
 Get(url string) (resp *http.Response, err error)
 Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
 error)
 Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
 scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
 return http.Get(url)
}
func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
 (resp *http.Response, err error) {
 putRequest, err := http.NewRequest("PUT", url, body)
 if err != nil {
  return nil, err
 }
 putRequest.ContentLength = contentLength
 return http.DefaultClient.Do(putRequest)
}
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
 {
 delRequest, err := http.NewRequest("DELETE", url, nil)
 if err != nil {
  return nil, err
 }
```

```
 return http.DefaultClient.Do(delRequest)
}
```

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

为对象生成预签名 URL，然后下载该 URL（GET 请求）。

导入。

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
```

```
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

生成 URL。

```
   /* Create a pre-signed URL to download an object in a subsequent GET request.
*/
   public String createPresignedGetUrl(String bucketName, String keyName) {
       try (S3Presigner presigner = S3Presigner.create()) {

           GetObjectRequest objectRequest = GetObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .build();

           GetObjectPresignRequest presignRequest =
GetObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL will
expire in 10 minutes.
                   .getObjectRequest(objectRequest)
                   .build();

           PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
           logger.info("Presigned URL: [{}]",
presignedRequest.url().toString());
           logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
   }
```

使用以下三种方法中的任何一种下载对象。

使用 JDK HttpURLConnection（自 v1.1 起）类进行下载。

```
   /* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
   public byte[] useHttpUrlConnectionToGet(String presignedUrlString) {
       ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

       try {
           URL presignedUrl = new URL(presignedUrlString);
           HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
           connection.setRequestMethod("GET");
           // Download the result of executing the request.
           try (InputStream content = connection.getInputStream()) {
               IoUtils.copy(content, byteArrayOutputStream);
           }
           logger.info("HTTP response code is " + connection.getResponseCode());

       } catch (S3Exception | IOException e) {
           logger.error(e.getMessage(), e);
       }
       return byteArrayOutputStream.toByteArray();
   }
```

使用 JDK HttpClient（自 v11 起）类进行下载。

```
   /* Use the JDK HttpClient (since v11) class to do the download. */
   public byte[] useHttpClientToGet(String presignedUrlString) {
       ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

       HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
       HttpClient httpClient = HttpClient.newHttpClient();
       try {
           URL presignedUrl = new URL(presignedUrlString);
           HttpResponse<InputStream> response = httpClient.send(requestBuilder
                           .uri(presignedUrl.toURI())
                           .GET()
                           .build(),
                   HttpResponse.BodyHandlers.ofInputStream());

           IoUtils.copy(response.body(), byteArrayOutputStream);
```

```
            logger.info("HTTP response code is " + response.statusCode());

        } catch (URISyntaxException | InterruptedException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

使用 AWS SDK for Java SdkHttpClient 类进行下载。

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToPut(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));
```

```
                logger.info("HTTP Response code is {}",
 response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

为上传生成预签名 URL，然后上传文件（PUT 请求）。

导入。

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
 software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
 software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
```

```
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

生成 URL。

```
   /* Create a presigned URL to use in a subsequent PUT request */
   public String createPresignedUrl(String bucketName, String keyName,
Map<String, String> metadata) {
       try (S3Presigner presigner = S3Presigner.create()) {

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .metadata(metadata)
                   .build();

           PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL
expires in 10 minutes.
                   .putObjectRequest(objectRequest)
                   .build();


           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           logger.info("Presigned URL to upload a file to: [{}]", myURL);
           logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

           return presignedRequest.url().toExternalForm();
       }
   }
```

使用以下三种方法中的任何一种上传文件对象。

使用 JDK HttpURLConnection（自 v1.1 起）类进行上传。

```
    /* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
    public void useHttpUrlConnectionToPut(String presignedUrlString, File
 fileToPut, Map<String, String> metadata) {
        logger.info("Begin [{}] upload", fileToPut.toString());
        try {
            URL presignedUrl = new URL(presignedUrlString);
            HttpURLConnection connection = (HttpURLConnection)
 presignedUrl.openConnection();
            connection.setDoOutput(true);
            metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
            connection.setRequestMethod("PUT");
            OutputStream out = connection.getOutputStream();

            try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
                 FileChannel inChannel = file.getChannel()) {
                ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
 8k

                while (inChannel.read(buffer) > 0) {
                    buffer.flip();
                    for (int i = 0; i < buffer.limit(); i++) {
                        out.write(buffer.get());
                    }
                    buffer.clear();
                }
            } catch (IOException e) {
                logger.error(e.getMessage(), e);
            }

            out.close();
            connection.getResponseCode();
            logger.info("HTTP response code is " + connection.getResponseCode());

        } catch (S3Exception | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

使用 JDK HttpClient（自 v11 起）类进行上传。

```
    /* Use the JDK HttpClient (since v11) class to do the upload. */
```

```
   public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
       metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

       HttpClient httpClient = HttpClient.newHttpClient();
       try {
           final HttpResponse<Void> response = httpClient.send(requestBuilder
                           .uri(new URL(presignedUrlString).toURI())

.PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI())))
                           .build(),
                   HttpResponse.BodyHandlers.discarding());

           logger.info("HTTP response code is " + response.statusCode());

       } catch (URISyntaxException | InterruptedException | IOException e) {
           logger.error(e.getMessage(), e);
       }
   }
```

使用 AWS for Java V2 SdkHttpClient 类进行上传。

```
   /* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
   public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
       logger.info("Begin [{}] upload", fileToPut.toString());

       try {
           URL presignedUrl = new URL(presignedUrlString);

           SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                   .method(SdkHttpMethod.PUT)
                   .uri(presignedUrl.toURI());
           // Add headers
           metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
k, v));
           // Finish building the request.
           SdkHttpRequest request = requestBuilder.build();
```

```
            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
```

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建预签名 URL 以将对象上传到存储桶。

```
import https from "https";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";
```

```
const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

function put(url, data) {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
        res.on("end", () => {
          resolve(responseBody);
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
    req.write(data);
    req.end();
  });
}

export const main = async () => {
```

```
  const REGION = "us-east-1";
  const BUCKET = "example_bucket";
  const KEY = "example_file.txt";

  // There are two ways to generate a presigned URL.
  // 1. Use createPresignedUrl without the S3 client.
  // 2. Use getSignedUrl in conjunction with the S3 client and GetObjectCommand.
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    const clientUrl = await createPresignedUrlWithClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (err) {
    console.error(err);
  }
};
```

创建预签名 URL 以从存储桶下载对象。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
```

```
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

export const main = async () => {
  const REGION = "us-east-1";
  const BUCKET = "example_bucket";
  const KEY = "example_file.jpg";

  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    const clientUrl = await createPresignedUrlWithClient({
      region: REGION,
      bucket: BUCKET,
      key: KEY,
    });

    console.log("Presigned URL without client");
    console.log(noClientUrl);
    console.log("\n");
```

```
    console.log("Presigned URL with client");
    console.log(clientUrl);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关更多信息，请参阅 AWS SDK for JavaScript 开发人员指南。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建 `GetObject` 预签名请求并使用 URL 下载对象。

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
 request to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()
```

```
    return objectContents
}
```

使用高级选项创建 GetObject 预签名请求。

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
 used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

创建 PutObject 预签名请求并使用它上传对象。

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
```

```
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
 HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }.put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- 有关更多信息，请参阅 AWS SDK for Kotlin 开发人员指南。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;
```

```
require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
 S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
 the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
 next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();
```

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

生成可在有限时间内执行 S3 操作的预签名 URL。使用请求软件包通过 URL 发出请求。

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
 ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
```

```
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method
        )
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to
 perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "r") as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
```

```
                f"Couldn't find {args.key}. For a PUT operation, the key must be
 the "
                f"name of a file that exists on your computer."
            )

    if response is not None:
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print("-" * 88)


if __name__ == "__main__":
    usage_demo()
```

生成预签名 POST 请求以上传文件。

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
 Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name


    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
 AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                 required access data.
```

```
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object
 '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response
```

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
require "aws-sdk-s3"
require "net/http"

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
```

```
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
 why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request("PUT", presigned_url.request_uri, object_content,
 "content_type" => "")
  end

  case response
  when Net::HTTPSuccess
    puts "Content uploaded!"
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建预签名请求以 GET 和 PUT S3 对象。

```
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());

    Ok(())
}

async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);

    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in)?)
        .await?;

    println!("Object URI: {}", presigned_request.uri());

    Ok(())
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建照片资产管理应用程序，让用户能够使用标签管理照片

以下代码示例演示如何创建无服务器应用程序，让用户能够使用标签管理照片。

.NET

AWS SDK for .NET

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK for C++

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

Java

SDK for Java 2.x

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

Kotlin

适用于 Kotlin 的 SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

适用于 PHP 的 SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

Rust

适用于 Rust 的 SDK

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

要深入了解这个例子的起源，请参阅 AWS 社区上的博文。

本示例中使用的服务

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 列出 Amazon S3 对象的网页

以下代码示例显示如何在网页中列出 Amazon S3 对象。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

以下代码是对 AWS SDK 进行调用的相关 React 组件。可以在前面的 GitHub 链接中找到包含此组件的应用程序的可运行版本。

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
 a role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
 traffic from
      // this example site. Here's an example configuration that allows all
 origins. Don't
      // do this in production.
      //[
      //  {
      //    "AllowedHeaders": ["*"],
      //    "AllowedMethods": ["GET"],
      //    "AllowedOrigins": ["*"],
      //    "ExposeHeaders": [],
      //  },
      //]
      //
      credentials: fromCognitoIdentityPool({
        clientConfig: { region: "us-east-1" },
        identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
      }),
```

```
    });
    const command = new ListObjectsCommand({ Bucket: "bucket-name" });
    client.send(command).then(({ Contents }) => setObjects(Contents || []));
  }, []);

  return (
    <div className="App">
      {objects.map((o) => (
        <div key={o.ETag}>{o.Key}</div>
      ))}
    </div>
  );
}

export default App;
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 ListObjects。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Amazon Textract 浏览器应用程序

以下代码示例展示如何通过交互式应用程序探索 Amazon Textract 输出。

JavaScript

SDK for JavaScript (v3)

演示了如何使用 AWS SDK for JavaScript 构建 React 应用程序，该应用程序使用 Amazon Textract 从文档图像中提取数据并在交互式网页中显示该数据。此示例在 Web 浏览器中运行，需要经过身份验证的 Amazon Cognito 身份才能获得凭证。它使用 Amazon Simple Storage Service（Amazon S3）进行存储；对于通知，它将轮询订阅 Amazon Simple Notification Service（Amazon SNS）主题的 Amazon Simple Queue Service（Amazon SQS）队列。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Cognito Identity

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

展示如何将 AWS SDK for Python (Boto3) 和 Amazon Textract 一起使用来检测文档图像中的文本、表单和表格元素。输入图像和 Amazon Textract 输出在 Tkinter 应用程序中显示，该应用程序可让您探索检测到的元素。

- 将文档图像提交到 Amazon Textract 并探索检测到的元素的输出。
- 将图像直接提交到 Amazon Textract，或通过 Amazon Simple Storage Service（Amazon S3）桶提交图像。
- 使用异步 API 启动任务，在任务完成后将通知发布到 Amazon Simple Notification Service (Amazon SNS) 主题。
- 轮询 Amazon Simple Queue Service (Amazon SQS) 队列，以获取任务完成消息并显示结果。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 删除未完成的 Amazon S3 分段上传

以下代码示例显示如何删除或停止未完成的 Amazon S3 分段上传。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

要停止正在进行或由于任何原因而未完成的分段上传，您可以获取上传列表，然后删除这些上传，如以下示例所示。

```
   public static void abortIncompleteMultipartUploadsFromList() {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
               .bucket(bucketName)
               .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                   .bucket(bucketName)
                   .key(upload.key())
                   .expectedBucketOwner(accountId)
                   .uploadId(upload.uploadId())
                   .build();

           AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
           if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
               logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
           }
       }
   }
```

要删除在某个日期之前或之后启动的未完成分段上传，您可以根据某个时间点有选择地删除分段上传，如以下示例所示。

```
   static void abortIncompleteMultipartUploadsOlderThan(Instant pointInTime) {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
               .bucket(bucketName)
               .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
           if (upload.initiated().isBefore(pointInTime)) {
               abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                       .bucket(bucketName)
                       .key(upload.key())
                       .expectedBucketOwner(accountId)
                       .uploadId(upload.uploadId())
                       .build();

               AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
               if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                   logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
               }
           }
       }
   }
```

如果您在开始分段上传后可以访问上传 ID，则可以使用该 ID 删除正在进行的上传。

```
   static void abortMultipartUploadUsingUploadId() {
       String uploadId = startUploadReturningUploadId();
```

```
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
                .uploadId(uploadId)
                .bucket(bucketName)
                .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
 uploadId, bucketName);
        }
    }
```

要一致地删除超过一定天数的未完成分段上传，请为存储桶设置存储桶生命周期配置。以下示例显示如何创建一条规则来删除超过 7 天的未完成上传。

```
    static void abortMultipartUploadsUsingLifecycleConfig() {
        Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
                .abortIncompleteMultipartUpload(b -> b.
                        daysAfterInitiation(7))
                .status("Enabled")
                .filter(SdkBuilder::build) // Filter element is required.
                .build());

        // If the action is successful, the service sends back an HTTP 200
 response with an empty HTTP body.
        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
                .bucket(bucketName)
                .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
 bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
 response.sdkHttpResponse().statusCode());
        }
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的以下主题。

- AbortMultipartUpload
- ListMultipartUploads
- PutBucketLifecycleConfiguration

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 通过 Amazon Rekognition 检测图像中的 PPE

以下代码示例展示如何构建采用 Amazon Rekognition 来检测图像中的个人防护设备 (PPE) 的应用程序。

Java

SDK for Java 2.x

展示如何创建 AWS Lambda 函数来检测包含个人防护设备的图像。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK for JavaScript (v3)

展示如何使用 Amazon Rekognition 和 AWS SDK for JavaScript 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的个人防护设备 (PPE)。该应用程序将结果保存到 Amazon DynamoDB 表，然后使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

了解如何：

- 使用 Amazon Cognito 创建未经身份验证的用户。

- 使用 Amazon Rekognition 分析包含 PPE 的图像。
- 为 Amazon SES 验证电子邮件地址。
- 使用结果更新 DynamoDB 表。
- 使用 Amazon SES 发送电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 检测从图像中提取的文本中的实体

以下代码示例显示了如何使用 Amazon Comprehend 检测 Amazon Textract 从存储在 Amazon S3 内的图像中提取的文本中的实体。

Python

SDK for Python (Boto3)

演示如何使用 AWS SDK for Python (Boto3) 在 Jupyter 笔记本中检测从图像中提取的文本中的实体。此示例使用 Amazon Textract 从存储在 Amazon Simple Storage Service (Amazon S3) 内的图像中提取文本，并使用 Amazon Comprehend 检测提取文本中的实体。

此示例是 Jupyter 笔记本，必须在可以托管笔记本电脑的环境中运行。有关如何使用 Amazon SageMaker 运行示例的说明，请参阅 TextractAndComprehendNotebook.ipynb 中的说明。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Comprehend
- Amazon S3

- Amazon Textract

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 检测图像中的人脸

以下代码示例展示了如何：

- 将图像保存到 Amazon S3 存储桶中。
- 使用 Amazon Rekognition 检测面部细节，例如年龄范围、性别和情绪（如微笑）。
- 显示这些细节。

Rust

适用于 Rust 的 SDK

将图像保存到具有 uploads 前缀的 Amazon S3 存储桶中，使用 Amazon Rekognition 检测面部细节，例如年龄范围、性别和情绪（微笑等），并显示这些细节。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS 软件开发工具包通过 Amazon Rekognition 检测图像中的对象

以下代码示例展示如何构建采用 Amazon Rekognition 来按类别检测图像中对象的应用程序。

.NET

AWS SDK for .NET

展示如何使用 Amazon Rekognition .NET API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像中

的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK for Java 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK for JavaScript (v3)

展示如何使用 Amazon Rekognition 和 AWS SDK for JavaScript 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

了解如何：

- 使用 Amazon Cognito 创建未经身份验证的用户。

- 使用 Amazon Rekognition 分析包含对象的图像。
- 为 Amazon SES 验证电子邮件地址。
- 使用 Amazon SES 发送电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

适用于 Kotlin 的 SDK

展示如何使用 Amazon Rekognition Kotlin API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK for Python (Boto3)

介绍如何使用 AWS SDK for Python (Boto3) 创建一个 Web 应用程序，让您可以执行以下操作：

- 将照片上载到 Amazon Simple Storage Service (Amazon S3) 存储桶。
- 使用 Amazon Rekognition 来分析和标注照片。
- 使用 Amazon Simple Email Service (Amazon SES) 发送图像分析的电子邮件报告。

此示例包含两个主要组件：用 JavaScript 编写的使用 React 构建的网页，以及用 Python 编写的使用 Flask-RESTful 构建的 REST 服务。

可以使用 React 网页执行以下操作：

- 显示存储在 S3 存储桶中的图像列表。
- 将计算机中的图像上载到 S3 存储桶。
- 显示图像和用于识别图像中检测到的物品的标注。
- 获取 S3 存储桶中所有图像的报告并发送报告电子邮件。

该网页调用 REST 服务。该服务将请求发送到 AWS 以执行以下操作：

- 获取并筛选 S3 存储桶中的图像列表。
- 将照片上载到 S3 存储桶。
- 使用 Amazon Rekognition 分析各张照片并获取标注列表，这些标注用于识别在照片中检测到的物品。
- 分析 S3 存储桶中的所有照片，然后使用 Amazon SES 通过电子邮件发送报告。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 通过 Amazon Rekognition 检测视频中的人物和对象

以下代码示例展示如何使用 Amazon Rekognition 检测视频中的人物和对象。

Java

SDK for Java 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK for JavaScript (v3)

展示如何使用 Amazon Rekognition 和 AWS SDK for JavaScript 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送包含结果的电子邮件通知。

了解如何：

- 使用 Amazon Cognito 创建未经身份验证的用户。
- 使用 Amazon Rekognition 分析包含 PPE 的图像。
- 为 Amazon SES 验证电子邮件地址。
- 使用 Amazon SES 发送电子邮件通知。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- Amazon SES

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 Amazon Simple Storage Service（Amazon S3）桶中的所有对象下载到本地目录

以下代码示例演示了如何将 Amazon Simple Storage Service（Amazon S3）桶中的所有对象下载到本地目录。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用 S3TransferManager 下载同一个 S3 存储桶中的所有 S3 对象。查看完整文件并进行测试。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
            URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
```

```
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 DownloadDirectory。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 从多区域接入点中获取 Amazon S3 对象

以下代码示例显示如何从多区域接入点中获取对象。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

将 S3 客户端配置为使用非对称 Sigv4（Sigv4a）签名算法。

```
        suspend fun createS3Client(): S3Client {
            // Configure your S3Client to use the Asymmetric Sigv4 (Sigv4a)
signing algorithm.
            val sigV4AScheme = SigV4AsymmetricAuthScheme(CrtAwsSigner)
            val s3 = S3Client.fromEnvironment {
                authSchemes = listOf(sigV4AScheme)
            }
            return s3
        }
```

使用多区域接入点 ARN 而不是桶名称来检索对象。

```
    suspend fun getObjectFromMrap(
```

```
        s3: S3Client,
        mrapArn: String,
        keyName: String,
    ): String? {
        val request = GetObjectRequest {
            bucket = mrapArn // Use the ARN instead of the bucket name for object
operations.
            key = keyName
        }

        var stringObj: String? = null
        s3.getObject(request) { resp ->
            stringObj = resp.body?.decodeToString()
            if (stringObj != null) {
                println("Successfully read $keyName from $mrapArn")
            }
        }
        return stringObj
    }
```

- 有关更多信息，请参阅 AWS SDK for Kotlin 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 GetObject。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 并指定 If-Modified-Since 标头以从 Amazon S3 存储桶获取对象

下面的代码示例显示如何从 S3 存储桶中的对象读取数据，但前提是该桶自上次检索以来未被修改。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
use aws_sdk_s3::{
    error::SdkError,
    operation::head_object::HeadObjectError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client, Error,
};
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{uuid}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
```

```
        .bucket(bucket_name.as_str())
        .key(KEY)
        .body(ByteStream::from_static(BODY.as_bytes()))
        .send()
        .await;

    // If the `PutObject` succeeded, get the eTag string from it. Otherwise,
    // report an error and return an empty string.
    let e_tag_1 = match put_object_output {
        Ok(put_object) => put_object.e_tag.unwrap(),
        Err(err) => {
            error!("{err:?}");
            String::new()
        }
    };

    // Request the object's headers.
    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await;

    // If the `HeadObject` request succeeded, create a tuple containing the
    // values of the headers `last-modified` and `etag`. If the request
    // failed, return the error in a tuple instead.
    let (last_modified, e_tag_2) = match head_object_output {
        Ok(head_object) => (
            Ok(head_object.last_modified().cloned().unwrap()),
            head_object.e_tag.unwrap(),
        ),
        Err(err) => (Err(err), String::new()),
    };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and first GetObject had differing eTags"
    );

    println!("First value of last_modified: {last_modified:?}");
    println!("First tag: {}\n", e_tag_1);
```

```
    // Send a second `HeadObject` request. This time, the `if_modified_since`
    // option is specified, giving the `last_modified` value returned by the
    // first call to `HeadObject`.
    //
    // Since the object hasn't been changed, and there are no other objects in
    // the bucket, there should be no matching objects.

    let head_object_output = client
        .head_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .if_modified_since(last_modified.unwrap())
        .send()
        .await;

    // If the `HeadObject` request succeeded, the result is a typle containing
    // the `last_modified` and `e_tag_1` properties. This is _not_ the expected
    // result.
    //
    // The _expected_ result of the second call to `HeadObject` is an
    // `SdkError::ServiceError` containing the HTTP error response. If that's
    // the case and the HTTP status is 304 (not modified), the output is a
    // tuple containing the values of the HTTP `last-modified` and `etag`
    // headers.
    //
    // If any other HTTP error occurred, the error is returned as an
    // `SdkError::ServiceError`.

    let (last_modified, e_tag_2): (Result<DateTime, SdkError<HeadObjectError>>,
String) =
        match head_object_output {
            Ok(head_object) => (
                Ok(head_object.last_modified().cloned().unwrap()),
                head_object.e_tag.unwrap(),
            ),
            Err(err) => match err {
                SdkError::ServiceError(err) => {
                    // Get the raw HTTP response. If its status is 304, the
                    // object has not changed. This is the expected code path.
                    let http = err.raw();
                    match http.status().as_u16() {
                        // If the HTTP status is 304: Not Modified, return a
                        // tuple containing the values of the HTTP
                        // `last-modified` and `etag` headers.
```

```
                        304 => (
                            Ok(DateTime::from_str(
                                http.headers().get("last-modified").unwrap(),
                                DateTimeFormat::HttpDate,
                            )
                            .unwrap()),
                            http.headers().get("etag").map(|t|
t.into()).unwrap(),
                        ),
                        // Any other HTTP status code is returned as an
                        // `SdkError::ServiceError`.
                        _ => (Err(SdkError::ServiceError(err)), String::new()),
                    }
                }
                // Any other kind of error is returned in a tuple containing the
                // error and an empty string.
                _ => (Err(err), String::new()),
            },
        };

    warn!("last modified: {last_modified:?}");
    assert_eq!(
        e_tag_1, e_tag_2,
        "PutObject and second HeadObject had different eTags"
    );

    println!("Second value of last modified: {last_modified:?}");
    println!("Second tag: {}", e_tag_2);

    // Clean up by deleting the object and the bucket.
    client
        .delete_object()
        .bucket(bucket_name.as_str())
        .key(KEY)
        .send()
        .await?;

    client
        .delete_bucket()
        .bucket(bucket_name.as_str())
        .send()
        .await?;

    Ok(())
```

```
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 GetObject。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 加密 Amazon S3 对象入门

下面的代码示例显示如何开始 Amazon S3 对象的加密。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";
```

```
            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

                // Download the object and verify that its contents match what
you uploaded.
                await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

                // Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

                // Copy both the source and target objects using server-side
encryption with
                // an encryption key.
                await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// Uploads an object to an Amazon S3 bucket.
        /// </summary>
```

```
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// PutObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket to which
 the
        /// object will be uploaded.</param>
        /// <param name="keyName">The name of the object to upload to the Amazon
 S3
        /// bucket.</param>
        /// <param name="base64Key">The encryption key.</param>
        /// <returns>The PutObjectRequest object for use by
 DownloadObjectAsync.</returns>
        public static async Task<PutObjectRequest> UploadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };
            PutObjectResponse putObjectResponse = await
 client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }

        /// <summary>
        /// Downloads an encrypted object from an Amazon S3 bucket.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
 call
        /// GetObjectAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
 object
        /// is located.</param>
        /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
        /// <param name="base64Key">The encryption key used to encrypt the
```

```
        /// object.</param>
        /// <param name="putObjectRequest">The PutObjectRequest used to upload
        /// the object.</param>
        public static async Task DownloadObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key,
            PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
                }

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same
as we set");
                }
                else
```

```
                {
                    Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
                }
            }
        }

        /// <summary>
        /// Retrieves the metadata associated with an Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used
        /// to call GetObjectMetadataAsync.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket containing
the
        /// object for which we want to retrieve metadata.</param>
        /// <param name="keyName">The name of the object for which we wish to
        /// retrieve the metadata.</param>
        /// <param name="base64Key">The encryption key associated with the
        /// object.</param>
        public static async Task GetObjectMetadataAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }
```

```
        /// <summary>
        /// Copies an encrypted object from one Amazon S3 bucket to another.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
call
        /// CopyObjectAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket containing the object
        /// to copy.</param>
        /// <param name="keyName">The name of the object to copy.</param>
        /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
        /// will be copied.</param>
        /// <param name="aesEncryption">The encryption type to use.</param>
        /// <param name="base64Key">The encryption key to use.</param>
        public static async Task CopyObjectAsync(
            IAmazonS3 client,
            string bucketName,
            string keyName,
            string copyTargetKeyName,
            Aes aesEncryption,
            string base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,

                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
            };
            await client.CopyObjectAsync(copyRequest);
        }
```

```
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for .NET API 参考中的以下主题。
  - CopyObject
  - GetObject
  - GetObjectMetadata

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 为 Amazon S3 存储桶和对象加标签入门

下面的代码示例显示如何开始为 AmazonS3 对象加标签。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
```

```
        {
            string bucketName = "doc-example-bucket";
            string keyName = "newobject.txt";
            string filePath = @"*** file path ***";

            // Specify your bucket region (an example region is shown).
            RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

            var client = new AmazonS3Client(bucketRegion);
            await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
        }

        /// <summary>
        /// This method uploads an object with tags. It then shows the tag
        /// values, changes the tags, and shows the new tags.
        /// </summary>
        /// <param name="client">The Initialized Amazon S3 client object used
        /// to call the methods to create and change an objects tags.</param>
        /// <param name="bucketName">A string representing the name of the
        /// bucket where the object will be stored.</param>
        /// <param name="keyName">A string representing the key name of the
        /// object to be tagged.</param>
        /// <param name="filePath">The directory location and file name of the
        /// object to be uploaded to the Amazon S3 bucket.</param>
        public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
        {
            try
            {
                // Create an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    FilePath = filePath,
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Keyx1", Value = "Value1" },
                        new Tag { Key = "Keyx2", Value = "Value2" },
                    },
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);
```

```
                // Now retrieve the new object's tags.
                GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

                // Display the tag values.
                objectTags.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

                Tagging newTagSet = new Tagging()
                {
                    TagSet = new List<Tag>
                    {
                        new Tag { Key = "Key3", Value = "Value3" },
                        new Tag { Key = "Key4", Value = "Value4" },
                    },
                };

                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet,
                };

                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // Retrieve the tags again and show the values.
                GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                };
```

```
                GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

                objectTags2.Tagging
                    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine(
                        $"Error: '{ex.Message}'");
            }
        }
    }
```

- 有关更多信息，请参阅《AWS SDK for .NET API 参考》中的 GetObjectTagging。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 获取 Amazon S3 存储桶的法定保留配置

以下代码示例演示如何获取 S3 存储桶的法定保留配置。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
```

```
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 GetObjectLegalHold。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
```

```
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 GetObjectLegalHold。

JavaScript

适用于 JavaScript 的 SDK（v3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import { fileURLToPath } from "url";
import { GetObjectLegalHoldCommand, S3Client } from "@aws-sdk/client-s3";

/**
 * @param {S3Client} client
 * @param {string} bucketName
 * @param {string} objectKey
```

```
 */
export const main = async (client, bucketName, objectKey) => {
  const command = new GetObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    // Optionally, you can provide additional parameters
    // ExpectedBucketOwner: "ACCOUNT_ID",
    // RequestPayer: "requester",
    // VersionId: "OBJECT_VERSION_ID",
  });

  try {
    const response = await client.send(command);
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (err) {
    console.error(err);
  }
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main(new S3Client(), "DOC-EXAMPLE-BUCKET", "OBJECT_KEY");
}
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 GetObjectLegalHold。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

通过 AWS SDK 使用 Amazon S3 对象锁定功能

以下代码示例显示了如何使用 S3 对象锁定特征。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

运行演示 Amazon S3 对象锁定功能的交互式场景。

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
        1. Create test Amazon Simple Storage Service (S3) buckets with different
 lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete
 or overwrite objects.
        5. Clean up objects and buckets.
   */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    private static string _resourcePrefix = null!;
```

```
    private static string noLockBucketName = null!;
    private static string lockEnabledBucketName = null!;
    private static string retentionAfterCreationBucketName = null!;
    private static List<string> bucketNames = new List<string>();
    private static List<string> fileNames = new List<string>();

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3ActionsWrapper>()
            )
            .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();

        ConfigurationSetup();

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Workflow Scenario.");
            Console.WriteLine(new string('-', 80));
            await Setup(true);

            await DemoActionChoices();

            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Object Locking Workflow is complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await Cleanup(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        noLockBucketName = _resourcePrefix + "-no-lock";
        lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
        retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-
creation";

        bucketNames.Add(noLockBucketName);
        bucketNames.Add(lockEnabledBucketName);
        bucketNames.Add(retentionAfterCreationBucketName);
    }

    // <summary>
```

```
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this workflow, we will use the AWS SDK for .NET to create
 several S3\n" +
            "buckets and files to demonstrate working with S3 locking features.
\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nS3 buckets can be created either with or without
 object lock enabled.");
        await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
 false);
        await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
 true);
        await
 _s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
 false);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nA bucket can be configured to use object locking
 with a default retention period.");
        await
 _s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
 true,
            ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("\nObject lock policies can also be added to existing
 buckets.");
```

```
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        // Upload some files to the buckets.
        Console.WriteLine("\nNow let's add some test files:");
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
        int fileCount = 2;
        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a bucket.");
        }

        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileCount; i++)
            {
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
                fileNames.Add(numberedFileName);
                await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
            }
        }
        Console.WriteLine("Press Enter to continue.");
        if (interactive)
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
            {
                // No modifications to the objects in the first bucket.
                if (bucketName != bucketNames[0])
                {
```

```
                    var exampleFileName = fileNames[i];
                    switch (i)
                    {
                        case 0:
                            {
                                var question =
                                    $"\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a legal hold.
                                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                                }
                                break;
                            }
                        case 1:
                            {
                                var question =
                                    $"\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                                    "\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a Governance mode retention period for
1 day.
                                    await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                                        bucketName, exampleFileName,
                                        ObjectLockRetentionMode.Governance,
                                        DateTime.UtcNow.AddDays(1));
                                }
                                break;
                            }
                    }
                }
            }
        }
        Console.WriteLine(new string('-', 80));
        return true;
```

```
    }

    // <summary>
    /// List all of the current buckets and objects.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>The list of buckets and objects.</returns>
    public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
    {
        var allObjects = new List<S3ObjectVersion>();
        foreach (var bucketName in bucketNames)
        {
            var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
            foreach (var objectKey in objectsInBucket.Versions)
            {
                allObjects.Add(objectKey);
            }
        }

        if (interactive)
        {
            Console.WriteLine("\nCurrent buckets and objects:\n");
            int i = 0;
            foreach (var bucketObject in allObjects)
            {
                i++;
                Console.WriteLine(
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{
```

```
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."};

        var choice = 0;
        // Keep asking the user until they choose to move on.
        while (choice != 6)
        {
            Console.WriteLine(new string('-', 80));
            choice = GetChoiceResponse(
                "\nExplore the S3 locking features by selecting one of the
following choices:"
                , choices);
            Console.WriteLine(new string('-', 80));
            switch (choice)
            {
                case 0:
                    {
                        await ListBucketsAndObjects(true);
                        break;
                    }
                case 1:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 2:
                    {
                        Console.WriteLine("\nEnter the number of the object to
delete:");
                        var allFiles = await ListBucketsAndObjects(true);
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
```

```
                        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
                        break;
                    }
                case 3:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object to
overwrite:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        // Create the file if it does not already exist.
                        if (!File.Exists(allFiles[fileChoice].Key))
                        {
                            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
                            await sw.WriteLineAsync(
                                "This is a sample file for uploading to a
bucket.");
                        }
                        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
                        break;
                    }
                case 4:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
                        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
                        break;
                    }
                case 5:
                    {
                        var allFiles = await ListBucketsAndObjects(true);
```

```
                        Console.WriteLine("\nEnter the number of the object to
view:");
                        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
                        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
                        break;
                    }
            }
        }
        return true;
    }

    // <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Cleanup(bool interactive)
    {
        Console.WriteLine(new string('-', 80));

        if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
        {
            // Remove all locks and delete all buckets and objects.
            var allFiles = await ListBucketsAndObjects(false);
            foreach (var fileInfo in allFiles)
            {
                // Check for a legal hold.
                var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
                if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
                {
                    await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
                }

                // Check for a retention period.
                var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
```

```
                var hasRetentionPeriod = retention?.Mode ==
 ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
 DateTime.UtcNow.Date;
                await
 _s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
 hasRetentionPeriod, fileInfo.VersionId);
            }

            foreach (var bucketName in bucketNames)
            {
                await _s3ActionsWrapper.DeleteBucketByName(bucketName);
            }

        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
```

```
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {choices[i]}");
            }
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }
}
```

S3 函数的包装器类。

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
```

```
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
 bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
 enableObjectLock)
    {
        Console.WriteLine($"\tCreating bucket {bucketName} with object lock
 {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }
```

```
    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
```

```
    /// Set or modify a retention period on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date retention expires.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
```

```
   public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
   {
       var enabledString = enableObjectLock ? "Enabled" : "Disabled";
       var timeDifference = retainUntilDate.Subtract(DateTime.Now);
       try
       {
           // First, enable Versioning on the bucket.
           await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
           {
               BucketName = bucketName,
               VersioningConfig = new S3BucketVersioningConfig()
               {
                   EnableMfaDelete = false,
                   Status = VersionStatus.Enabled
               }
           });

           var request = new PutObjectLockConfigurationRequest()
           {
               BucketName = bucketName,
               ObjectLockConfiguration = new ObjectLockConfiguration()
               {
                   ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                   Rule = new ObjectLockRule()
                   {
                       DefaultRetention = new DefaultRetention()
                       {
                           Mode = retention,
                           Days = timeDifference.Days // Can be specified in
days or years but not both.
                       }
                   }
               }
           };

           var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
           Console.WriteLine($"\tAdded a default retention to bucket
{bucketName}.");
           return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
       }
       catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tError modifying object lock: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the retention period for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\tObject retention for {objectKey} in
{bucketName}: " +
                              $"\n\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
```

```
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };
```

```
            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\tObject legal hold for {objectKey} in
{bucketName}: " +
                              $"\n\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
    {
        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\tBucket object lock config for {bucketName} in
{bucketName}: " +
                              $"\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                              $"\n\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to fetch object lock config:
'{ex.Message}'");
```

```
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.<returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };

        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\tSuccessfully uploaded {objectName} to
{bucketName}.");
            return true;
        }
        else
        {
            Console.WriteLine($"\tCould not upload {objectName} to
{bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
```

```
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
 bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
 objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
```

```
        {
            Console.WriteLine($"\tUnable to delete object {objectKey} in bucket
 {bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\tUnable to delete bucket {bucketName}: " +
 ex.Message);
            return false;
        }

    }

}
```

- 有关 API 详细信息，请参阅 AWS SDK for .NET API 参考中的以下主题。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration

- PutObjectRetention

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

运行演示 Amazon S3 对象锁定功能的交互式场景。

```
// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
 questioner demotools.IQuestioner
 resources  Resources
 s3Actions  *actions.S3Actions
 sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner) ObjectLockScenario {
 scenario := ObjectLockScenario{
  questioner: questioner,
  resources:  Resources{},
  s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
  sdkConfig:  sdkConfig,
 }
 scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
 scenario.resources.init(scenario.s3Actions, questioner)
 return scenario
}

type nameLocked struct {
 name   string
 locked bool
}
```

```
var createInfo = []nameLocked{
 {"standard-bucket", false},
 {"lock-bucket", true},
 {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
 log.Println("Let's create some S3 buckets to use for this workflow.")
 success := false
 for !success {
  prefix := scenario.questioner.Ask(
   "This example creates three buckets. Enter a prefix to name your buckets
 (remember bucket names must be globally unique):")

  for _, info := range createInfo {
   bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
 fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
   if err != nil {
    switch err.(type) {
    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
     log.Printf("Couldn't create bucket %s.\n", bucketName)
    default:
     panic(err)
    }
    break
   }
   scenario.resources.demoBuckets[info.name] = &DemoBucket{
    name:       bucketName,
    objectKeys: []string{},
   }
   log.Printf("Created bucket %s.\n", bucketName)
  }

  if len(scenario.resources.demoBuckets) < len(createInfo) {
   scenario.resources.deleteBuckets(ctx)
  } else {
   success = true
  }
 }

 log.Println("S3 buckets created.")
 log.Println(strings.Repeat("-", 88))
}
```

```
// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
 log.Println("\nA bucket can be configured to use object locking.")
 scenario.questioner.Ask("Press Enter to continue.")

 var err error
 bucket := scenario.resources.demoBuckets["retention-bucket"]
 err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
  default:
   panic(err)
  }
 } else {
  log.Printf("Object locking enabled on bucket %s.", bucket.name)
 }

 log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
 bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
 context.Context) {
 log.Println("\nA bucket can be configured to use object locking with a default
 retention period.")

 bucket := scenario.resources.demoBuckets["retention-bucket"]
 retentionPeriod := scenario.questioner.AskInt("Enter the default retention
 period in days: ")
 err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
 bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
 types.ObjectLockRetentionModeGovernance)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Printf("Couldn't configure a default retention period on bucket %s.\n",
 bucket.name)
  default:
   panic(err)
  }
```

```
 } else {
  log.Printf("Default retention policy set on bucket %s with %d day retention
 period.", bucket.name, retentionPeriod)
  bucket.retentionEnabled = true
 }

 log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
 log.Println("Uploading test objects to S3 buckets.")

 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
  for i := 0; i < 2; i++ {
   key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
 fmt.Sprintf("example-%d", i),
    fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
   if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
     log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
    default:
     panic(err)
    }
   } else {
    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
   }
  }
 }

 scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
 log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
 objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
 context.Context) {
 log.Println("Now let's set object lock configurations on individual objects.")
```

```
buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
scenario.resources.demoBuckets["retention-bucket"]}
for _, bucket := range buckets {
 for index, objKey := range bucket.objectKeys {
  switch index {
  case 0:
   if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
to %s in %s (y/n)? ", objKey, bucket.name), "y") {
    err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
types.ObjectLockLegalHoldStatusOn)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Couldn't set legal hold on %s.\n", objKey)
     default:
      panic(err)
     }
    } else {
     log.Printf("Legal hold set on %s.\n", objKey)
    }
   }
  case 1:
   q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
%s in %s?\n"+
    "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
able to delete this object\n"+
    "or its bucket until the retention period has expired. (y/n) ", objKey,
bucket.name)
   if scenario.questioner.AskBool(q, "y") {
    err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
types.ObjectLockRetentionModeGovernance, 1)
    if err != nil {
     switch err.(type) {
     case *types.NoSuchKey:
      log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
bucket.name)
     default:
      panic(err)
     }
    } else {
     log.Printf("Retention period set to 1 for %s.", objKey)
     bucket.retentionEnabled = true
    }
   }
```

```
   }
  }
 }
 log.Println(strings.Repeat("-", 88))
}

const (
 ListAll = iota
 DeleteObject
 DeleteRetentionObject
 OverwriteObject
 ViewRetention
 ViewLegalHold
 Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
 object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
 log.Println("Now you can interact with the objects to explore the object lock
 configurations.")
 interactiveChoices := []string{
  "List all objects and buckets.",
  "Attempt to delete an object.",
  "Attempt to delete an object with retention period bypass.",
  "Attempt to overwrite a file.",
  "View the retention settings for an object.",
  "View the legal hold settings for an object.",
  "Finish the workflow."}

 choice := ListAll
 for choice != Finish {
  objList := scenario.GetAllObjects(ctx)
  objChoices := scenario.makeObjectChoiceList(objList)
  choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
 interactiveChoices)
  switch choice {
  case ListAll:
   log.Println("The current objects in the example buckets are:")
   for _, objChoice := range objChoices {
    log.Println("\t", objChoice)
   }
  case DeleteObject, DeleteRetentionObject:
```

```
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
  obj := objList[objChoice]
  deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Println("Nothing to delete.")
   default:
    panic(err)
   }
  } else if deleted {
   log.Printf("Object %s deleted.\n", obj.key)
  }
 case OverwriteObject:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
  obj := objList[objChoice]
  _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Println("Couldn't upload to nonexistent bucket.")
   default:
    panic(err)
   }
  } else {
   log.Printf("Uploaded new content to object %s.\n", obj.key)
  }
 case ViewRetention:
  objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
  obj := objList[objChoice]
  retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Printf("Can't get retention configuration for %s.\n", obj.key)
   default:
    panic(err)
   }
```

```
   } else if retention != nil {
    log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
 retention.Mode, retention.RetainUntilDate)
   } else {
    log.Printf("Object %s does not have object retention configured.\n", obj.key)
   }
  case ViewLegalHold:
   objChoice := scenario.questioner.AskChoice("Enter the number of the object to
 view:\n", objChoices)
   obj := objList[objChoice]
   legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
 obj.key, obj.versionId)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
    default:
     panic(err)
    }
   } else if legalHold != nil {
    log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
   } else {
    log.Printf("Object %s does not have legal hold configured.", obj.key)
   }
  case Finish:
   log.Println("Let's clean up.")
  }
  log.Println(strings.Repeat("-", 88))
 }
}

type BucketKeyVersionId struct {
 bucket    string
 key       string
 versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
 them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
 []BucketKeyVersionId {
 var objectList []BucketKeyVersionId
 for _, info := range createInfo {
  bucket := scenario.resources.demoBuckets[info.name]
```

```
  versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchBucket:
    log.Printf("Couldn't get object versions for %s.\n", bucket.name)
   default:
    panic(err)
   }
  } else {
   for _, version := range versions {
    objectList = append(objectList,
     BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
 *version.VersionId})
   }
  }
 }
 return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
 are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
 []BucketKeyVersionId) []string {
 choices := make([]string, len(bucketObjects))
 for i := 0; i < len(bucketObjects); i++ {
  choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
   bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
 }
 return choices
}

// Run runs the S3 Object Lock workflow scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   _, isMock := scenario.questioner.(*demotools.MockQuestioner)
   if isMock || scenario.questioner.AskBool("Do you want to see the full error
 message (y/n)?", "y") {
    log.Println(r)
   }
   scenario.resources.Cleanup(ctx)
  }
```

```
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon S3 Object Lock Workflow Scenario.")
 log.Println(strings.Repeat("-", 88))

 scenario.CreateBuckets(ctx)
 scenario.EnableLockOnBucket(ctx)
 scenario.SetDefaultRetentionPolicy(ctx)
 scenario.UploadTestObjects(ctx)
 scenario.SetObjectLockConfigurations(ctx)
 scenario.InteractWithObjects(ctx)

 scenario.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

定义一个结构来包装此示例中使用的 S3 操作。

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
 S3Client  *s3.Client
 S3Manager *manager.Uploader
}



// CreateBucketWithLock creates a new S3 bucket with optional object locking
 enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
 region string, enableObjectLock bool) (string, error) {
 input := &s3.CreateBucketInput{
  Bucket: aws.String(bucket),
  CreateBucketConfiguration: &types.CreateBucketConfiguration{
   LocationConstraint: types.BucketLocationConstraint(region),
  },
```

```
 }

 if enableObjectLock {
  input.ObjectLockEnabledForBucket = aws.Bool(true)
 }

 _, err := actor.S3Client.CreateBucket(ctx, input)
 if err != nil {
  var owned *types.BucketAlreadyOwnedByYou
  var exists *types.BucketAlreadyExists
  if errors.As(err, &owned) {
   log.Printf("You already own bucket %s.\n", bucket)
   err = owned
  } else if errors.As(err, &exists) {
   log.Printf("Bucket %s already exists.\n", bucket)
   err = exists
  }
 } else {
  err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
   ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
  }
 }

 return bucket, err
}



// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
 var status *types.ObjectLockLegalHoldStatus
 input := &s3.GetObjectLegalHoldInput{
  Bucket:    aws.String(bucket),
  Key:       aws.String(key),
  VersionId: aws.String(versionId),
 }

 output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
 if err != nil {
  var noSuchKeyErr *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
```

```
  if errors.As(err, &noSuchKeyErr) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noSuchKeyErr
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    log.Printf("Object %s does not have an object lock configuration.\n", key)
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
   }
  }
 } else {
  status = &output.LegalHold.Status
 }

 return status, err
}



// GetObjectLockConfiguration retrieves the object lock configuration for an S3
 bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
 string) (*types.ObjectLockConfiguration, error) {
 var lockConfig *types.ObjectLockConfiguration
 input := &s3.GetObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
 }

 output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
 "ObjectLockConfigurationNotFoundError" {
   log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
   err = nil
  }
 } else {
```

```
  lockConfig = output.ObjectLockConfiguration
 }

 return lockConfig, err
}



// GetObjectRetention retrieves the object retention configuration for an S3
 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
 string) (*types.ObjectLockRetention, error) {
 var retention *types.ObjectLockRetention
 input := &s3.GetObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }

 output, err := actor.S3Client.GetObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "NoSuchObjectLockConfiguration":
    err = nil
   case "InvalidRequest":
    log.Printf("Bucket %s does not have locking enabled.", bucket)
    err = nil
   }
  }
 } else {
  retention = output.Retention
 }

 return retention, err
}



// PutObjectLegalHold sets the legal hold configuration for an S3 object.
```

```
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
 string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
 {
 input := &s3.PutObjectLegalHoldInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  LegalHold: &types.ObjectLockLegalHold{
   Status: legalHoldStatus,
  },
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }

 _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// ModifyDefaultBucketRetention modifies the default retention period of an
 existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
 ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
 retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: lockMode,
   Rule: &types.ObjectLockRule{
    DefaultRetention: &types.DefaultRetention{
     Days: aws.Int32(retentionPeriod),
     Mode: retentionMode,
    },
   },
```

```
  },
 }
 _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
 string) error {
 // Versioning must be enabled on the bucket before object locking is enabled.
 verInput := &s3.PutBucketVersioningInput{
  Bucket: aws.String(bucket),
  VersioningConfiguration: &types.VersioningConfiguration{
   MFADelete: types.MFADeleteDisabled,
   Status:    types.BucketVersioningStatusEnabled,
  },
 }
 _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
  return err
 }

 input := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucket),
  ObjectLockConfiguration: &types.ObjectLockConfiguration{
   ObjectLockEnabled: types.ObjectLockEnabledEnabled,
  },
 }
 _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
```

```
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 }

 return err
}



// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
 string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
 error {
 input := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
  Retention: &types.ObjectLockRetention{
   Mode:            retentionMode,
   RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
  },
  BypassGovernanceRetention: aws.Bool(true),
 }

 _, err := actor.S3Client.PutObjectRetention(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
   err = noKey
  }
 }

 return err
}



// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
 string, contents string) (string, error) {
```

```
 var outKey string
 input := &s3.PutObjectInput{
  Bucket:            aws.String(bucket),
  Key:               aws.String(key),
  Body:              bytes.NewReader([]byte(contents)),
  ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
 }
 output, err := actor.S3Manager.Upload(ctx, input)
 if err != nil {
  var noBucket *types.NoSuchBucket
  if errors.As(err, &noBucket) {
   log.Printf("Bucket %s does not exist.\n", bucket)
   err = noBucket
  }
 } else {
  err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
   Bucket: aws.String(bucket),
   Key:    aws.String(key),
  }, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
 bucket)
  } else {
   outKey = *output.Key
  }
 }
 return outKey, err
}



// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
 ([]types.ObjectVersion, error) {
 var err error
 var output *s3.ListObjectVersionsOutput
 var versions []types.ObjectVersion
 input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
 versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
 for versionPaginator.HasMorePages() {
  output, err = versionPaginator.NextPage(ctx)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
```

```
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
   break
  } else {
   versions = append(versions, output.Versions...)
  }
 }
 return versions, err
}



// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
 string, versionId string, bypassGovernance bool) (bool, error) {
 deleted := false
 input := &s3.DeleteObjectInput{
  Bucket: aws.String(bucket),
  Key:    aws.String(key),
 }
 if versionId != "" {
  input.VersionId = aws.String(versionId)
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 _, err := actor.S3Client.DeleteObject(ctx, input)
 if err != nil {
  var noKey *types.NoSuchKey
  var apiErr *smithy.GenericAPIError
  if errors.As(err, &noKey) {
   log.Printf("Object %s does not exist in %s.\n", key, bucket)
   err = noKey
  } else if errors.As(err, &apiErr) {
   switch apiErr.ErrorCode() {
   case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
   case "InvalidArgument":
    if bypassGovernance {
     log.Printf("You cannot specify bypass governance on a bucket without lock
 enabled.")
     err = nil
```

```
    }
   }
  }
 } else {
  deleted = true
 }
 return deleted, err
}



// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
 []types.ObjectIdentifier, bypassGovernance bool) error {
 if len(objects) == 0 {
  return nil
 }

 input := s3.DeleteObjectsInput{
  Bucket: aws.String(bucket),
  Delete: &types.Delete{
   Objects: objects,
   Quiet:   aws.Bool(true),
  },
 }
 if bypassGovernance {
  input.BypassGovernanceRetention = aws.Bool(true)
 }
 delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
 if err != nil || len(delOut.Errors) > 0 {
  log.Printf("Error deleting objects from bucket %s.\n", bucket)
  if err != nil {
   var noBucket *types.NoSuchBucket
   if errors.As(err, &noBucket) {
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
   }
  } else if len(delOut.Errors) > 0 {
   for _, outErr := range delOut.Errors {
    log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
   }
   err = fmt.Errorf("%s", *delOut.Errors[0].Message)
  }
 }
```

```
 return err
}
```

清理资源。

```
// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
 name             string
 legalHold        bool
 retentionEnabled bool
 objectKeys       []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
 and handles
// cleanup when the scenario finishes.
type Resources struct {
 demoBuckets map[string]*DemoBucket

 s3Actions  *actions.S3Actions
 questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
 demotools.IQuestioner) {
 resources.s3Actions = s3Actions
 resources.questioner = questioner
 resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 " +
    "that were created for this scenario.")
```

```
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if !wantDelete {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
  return
 }

 log.Println("Removing objects from S3 buckets and deleting buckets...")
 resources.deleteBuckets(ctx)
 //resources.deleteRetentionObjects(resources.retentionBucket,
 resources.retentionObjects)

 log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
 ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
 for _, info := range createInfo {
  bucket := resources.demoBuckets[info.name]
  resources.deleteObjects(ctx, bucket)
  _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
   Bucket: aws.String(bucket.name),
  })
  if err != nil {
   panic(err)
  }
 }
 resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
 *DemoBucket) {
 lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
 bucket.name)
 if err != nil {
  panic(err)
 }
```

```
versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
if err != nil {
 switch err.(type) {
 case *types.NoSuchBucket:
  log.Printf("No objects to get from %s.\n", bucket.name)
 default:
  panic(err)
 }
}
delObjects := make([]types.ObjectIdentifier, len(versions))
for i, version := range versions {
 if lockConfig != nil && lockConfig.ObjectLockEnabled ==
types.ObjectLockEnabledEnabled {
  status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)
  if err != nil {
   switch err.(type) {
   case *types.NoSuchKey:
    log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
   default:
    panic(err)
   }
  } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
   err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
   if err != nil {
    switch err.(type) {
    case *types.NoSuchKey:
     log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
    default:
     panic(err)
    }
   }
  }
 }
 delObjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, delObjects,
bucket.retentionEnabled)
if err != nil {
 switch err.(type) {
```

```
  case *types.NoSuchBucket:
   log.Println("Nothing to delete.")
  default:
   panic(err)
  }
 }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的以下主题。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

运行演示 Amazon S3 对象锁定功能的交互式场景。

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
```

```
import java.util.stream.Collectors;

/*
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        // Get the current date and time to ensure bucket name is unique.
        LocalDateTime currentTime = LocalDateTime.now();

        // Format the date and time as a string.
        DateTimeFormatter formatter =
 DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        String timeStamp = currentTime.format(formatter);

        s3LockActions = new S3LockActions();
        bucketName = "bucket"+timeStamp;
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
 Object Locking Workflow Scenario.");
        System.out.println("Press Enter to continue...");
```

```
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
```

```
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
                    System.out.println("Enter the number of the object to
delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
```

```
                }

                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }

                case 4 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to
view:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
```

```
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }

    // Clean up the resources from the scenario.
    private static void cleanup() {
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
        for (S3InfoObject fileInfo : allFiles) {
            String bucketName = fileInfo.getBucketName();
            String key = fileInfo.getKeyName();
            String version = fileInfo.getVersion();
            if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
                ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
                if (legalHold != null) {
                    String holdStatus = legalHold.status().name();
                    System.out.println(holdStatus);
                    if (holdStatus.compareTo("ON") == 0) {
                        s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                    }
                }
                // Check for a retention period.
```

```
                ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
                boolean hasRetentionPeriod ;
                hasRetentionPeriod = retention != null;
                s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

            } else {
                System.out.println(bucketName +" objects do not have a legal
lock");
                s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
            }
        }

        // Delete the buckets.
        System.out.println("Delete "+bucketName);
        for (String bucket : bucketNames){
            s3LockActions.deleteBucketByName(bucket);
        }
    }

    private static void setup() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
several S3
                buckets and files to demonstrate working with S3 locking
features.
                """);

        System.out.println("S3 buckets can be created either with or without
object lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();
```

```
        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
```

```
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on
individual files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
                        }
                        case 1 -> {
                            """
                                Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
                                Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
                                """.formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }
```

```
    }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }

        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid
 number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

S3 函数的包装器类。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
 software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
 {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time
 zone.
        ZonedDateTime zonedDateTime =
 futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
 HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();
```

```
        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
```

```
            .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                }))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": "+ s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
```

```
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        getClient().putBucketVersioning(versioningRequest);
        DefaultRetention rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();


getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
```

```
        System.out.println("Added a default retention to bucket "+bucketName
+".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath)
{
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
```

```
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
```

```
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
        }
    }
```

```
    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
 GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
}
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的以下主题。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

JavaScript

适用于 JavaScript 的 SDK（v3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

index.js - 工作流的入口点。这用于编排所有步骤。访问 GitHub，来查看 Scenario、ScenarioInput、ScenarioOutput 和 ScenarioAction 的实现详细信息。

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "./welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
```

```
} from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
```

```
    demo: new scenarios.Scenario(
      "S3 Object Locking - Demo",
      [loadState, replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Object Locking - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};

// Call function if run directly
import { fileURLToPath } from "url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios);
}
```

welcome.steps.js - 向控制台输出欢迎消息。

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
```

```
    `Welcome to the Amazon Simple Storage Service (S3) Object Locking Workflow
 Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
 several S3 buckets and files to demonstrate working with S3 locking features.`,
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

setup.steps.js - 部署存储桶、对象和文件设置。

```
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */
```

```
const bucketPrefix = "js-object-locking";

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    `The following buckets will be created:
         ${bucketPrefix}-no-lock with object lock False.
         ${bucketPrefix}-lock-enabled with object lock True.
         ${bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${bucketPrefix}-retention-after-creation`;

    await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
    await client.send(
      new CreateBucketCommand({
        Bucket: lockEnabledBucketName,
        ObjectLockEnabledForBucket: true,
      }),
    );
    await client.send(new CreateBucketCommand({ Bucket: retentionBucketName }));

    state.noLockBucketName = noLockBucketName;
```

```
    state.lockEnabledBucketName = lockEnabledBucketName;
    state.retentionBucketName = retentionBucketName;
  });

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    `The following test files will be created:
         file0.txt in ${bucketPrefix}-no-lock.
         file1.txt in ${bucketPrefix}-no-lock.
         file0.txt in ${bucketPrefix}-lock-enabled.
         file1.txt in ${bucketPrefix}-lock-enabled.
         file0.txt in ${bucketPrefix}-retention-after-creation.
         file1.txt in ${bucketPrefix}-retention-after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    await client.send(
      new PutObjectCommand({
        Bucket: state.noLockBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
```

```
    await client.send(
      new PutObjectCommand({
        Bucket: state.noLockBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.lockEnabledBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.lockEnabledBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
    await client.send(
      new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
```

```
 */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    `A bucket can be configured to use object locking with a default retention
 period.
   A default retention period will be configured for ${bucketPrefix}-retention-
after-creation.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );

    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.retentionBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
          Rule: {
            DefaultRetention: {
```

```
              Mode: "GOVERNANCE",
              Years: 1,
            },
          },
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    `Object lock policies can also be added to existing buckets.
   An object lock policy will be added to ${bucketPrefix}-lock-enabled.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateLockPolicy",
    "Add object lock policy?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
```

```
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
```

```
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.lockEnabledBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.lockEnabledBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
  );

/**
 * @param {Scenarios} scenarios
```

```
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
 ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
```

```
      `Would you like to add a 1 day Governance retention period to file1.txt in
 ${state.retentionBucketName}?
Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
 able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        }),
      );
      console.log(
        `Set retention for file1.txt in ${state.retentionBucketName} until
 ${retentionDate.toISOString().split("T")[0]}.`,
      );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
  );

export {
  createBuckets,
  confirmCreateBuckets,
  createBucketsAction,
  populateBuckets,
  confirmPopulateBuckets,
```

```
  populateBucketsAction,
  updateRetention,
  confirmUpdateRetention,
  updateRetentionAction,
  updateLockPolicy,
  confirmUpdateLockPolicy,
  updateLockPolicyAction,
  confirmSetLegalHoldFileEnabled,
  setLegalHoldFileEnabledAction,
  confirmSetRetentionPeriodFileEnabled,
  setRetentionPeriodFileEnabledAction,
  confirmSetLegalHoldFileRetention,
  setLegalHoldFileRetentionAction,
  confirmSetRetentionPeriodFileRetention,
  setRetentionPeriodFileRetentionAction,
};
```

repl.steps.js - 查看和删除存储桶中的文件。

```
import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
```

```
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    `Explore the S3 locking features by selecting one of the following choices`,
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
```

```
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;
      files.push({ bucket, key: Key, version: VersionId });
    }
  }

  return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.noLockBucketName,
        state.lockEnabledBucketName,
        state.retentionBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const { replChoice } = state;

      switch (replChoice) {
        case choices.LIST_ALL_FILES: {
          const files = await getAllFiles(client, [
```

```
            state.noLockBucketName,
            state.lockEnabledBucketName,
            state.retentionBucketName,
          ]);
          state.replOutput = files
            .map(
              (file) =>
                `${file.bucket}: ${file.key} (version: ${file.version})`,
            )
            .join("\n");
          break;
        }
        case choices.DELETE_FILE: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.DELETE_FILE_WITH_RETENTION: {
          /** @type {number} */
          const fileToDelete = await fileInput.handle(state);
          const selectedFile = files[fileToDelete];
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
                BypassGovernanceRetention: true,
              }),
```

```
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.OVERWRITE_FILE: {
          /** @type {number} */
          const fileToOverwrite = await fileInput.handle(state);
          const selectedFile = files[fileToOverwrite];
          try {
            await client.send(
              new PutObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
              }),
            );
            state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
          } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
          }
          break;
        }
        case choices.VIEW_RETENTION_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const retention = await client.send(
              new GetObjectRetentionCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            const bucketConfig = await client.send(
              new GetObjectLockConfigurationCommand({
```

```
                Bucket: selectedFile.bucket,
              }),
            );
            state.replOutput = `Object retention for ${selectedFile.key}
 in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
 ${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
 ${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
          } catch (err) {
            state.replOutput = `Unable to fetch object lock retention:
 '${err.message}'`;
          }
          break;
        }
        case choices.VIEW_LEGAL_HOLD_SETTINGS: {
          /** @type {number} */
          const fileToView = await fileInput.handle(state);
          const selectedFile = files[fileToView];
          try {
            const legalHold = await client.send(
              new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
              }),
            );
            state.replOutput = `Object legal hold for ${selectedFile.key} in
 ${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
          } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
          }
          break;
        }
        default:
          throw new Error(`Invalid replChoice: ${replChoice}`);
      }
    },
    {
      whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
```

```
          "REPL output",
          (state) => state.replOutput,
          { preformatted: true },
        ),
      },
    },
  );

export { replInput, replAction, choices };
```

clean.steps.js - 销毁所有创建的资源。

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,
  PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
```

```
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
        } else {
          throw e;
        }
      }

      for (const version of objectsResponse.Versions || []) {
        const { Key, VersionId } = version;

        try {
          const legalHold = await client.send(
            new GetObjectLegalHoldCommand({
              Bucket: bucket,
              Key,
              VersionId,
            }),
          );

          if (legalHold.LegalHold?.Status === "ON") {
            await client.send(
              new PutObjectLegalHoldCommand({
```

```
                Bucket: bucket,
                Key,
                VersionId,
                LegalHold: {
                  Status: "OFF",
                },
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch legal hold for ${Key} in ${bucket}:
'${err.message}'`,
          );
        }

        try {
          const retention = await client.send(
            new GetObjectRetentionCommand({
              Bucket: bucket,
              Key,
              VersionId,
            }),
          );

          if (retention.Retention?.Mode === "GOVERNANCE") {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
                BypassGovernanceRetention: true,
              }),
            );
          }
        } catch (err) {
          console.log(
            `Unable to fetch object lock retention for ${Key} in ${bucket}:
'${err.message}'`,
          );
        }

        await client.send(
          new DeleteObjectCommand({
```

```
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );
      }

      await client.send(new DeleteBucketCommand({ Bucket: bucket }));
      console.log(`Delete for ${bucket} complete.`);
    }
  });

export { confirmCleanup, cleanupAction };
```

- 有关 API 详细信息，请参阅 AWS SDK for JavaScript API 参考中的以下主题。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 管理 Amazon S3 存储桶的访问控制列表（ACL）

以下代码示例显示如何管理 Amazon S3 存储桶的访问控制列表（ACL）。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   using System;
   using System.Collections.Generic;
   using System.Threading.Tasks;
   using Amazon.S3;
   using Amazon.S3.Model;

   /// <summary>
   /// This example shows how to manage Amazon Simple Storage Service
   /// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
   /// access.
   /// </summary>
   public class ManageACLs
   {
       public static async Task Main()
       {
           string bucketName = "doc-example-bucket1";
           string newBucketName = "doc-example-bucket2";
           string keyName = "sample-object.txt";
           string emailAddress = "someone@example.com";

           // If the AWS Region where your bucket is located is different from
           // the Region defined for the default user, pass the Amazon S3
bucket's
           // name to the client constructor. It should look like this:
           // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
           IAmazonS3 client = new AmazonS3Client();

           await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
       }

       /// <summary>
       /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
       /// information and then adds a new ACL to one of the objects in the
       /// Amazon S3 bucket.
       /// </summary>
       /// <param name="client">The initialized Amazon S3 client object used to
call
       /// methods to create a bucket, get an ACL, and add a different ACL to
       /// one of the objects.</param>
       /// <param name="bucketName">A string representing the original Amazon S3
       /// bucket name.</param>
```

```
        /// <param name="newBucketName">A string representing the name of the
        /// new bucket that will be created.</param>
        /// <param name="keyName">A string representing the key name of an Amazon
 S3
        /// object for which we will change the ACL.</param>
        /// <param name="emailAddress">A string representing the email address
        /// belonging to the person to whom access to the Amazon S3 bucket will
 be
        /// granted.</param>
        public static async Task TestBucketObjectACLsAsync(
            IAmazonS3 client,
            string bucketName,
            string newBucketName,
            string keyName,
            string emailAddress)
        {
            try
            {
                // Create a new Amazon S3 bucket and specify canned ACL.
                var success = await CreateBucketWithCannedACLAsync(client,
 newBucketName);

                // Get the ACL on a bucket.
                await GetBucketACLAsync(client, bucketName);

                // Add (replace) the ACL on an object in a bucket.
                await AddACLToExistingObjectAsync(client, bucketName, keyName,
 emailAddress);
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine($"Exception: {amazonS3Exception.Message}");
            }
        }

        /// <summary>
        /// Creates a new Amazon S3 bucket with a canned ACL attached.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="newBucketName">A string representing the name of the
        /// new Amazon S3 bucket.</param>
        /// <returns>Returns a boolean value indicating success or failure.</
returns>
```

```
        public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
client, string newBucketName)
        {
            var request = new PutBucketRequest()
            {
                BucketName = newBucketName,
                BucketRegion = S3Region.EUWest1,

                // Add a canned ACL.
                CannedACL = S3CannedACL.LogDeliveryWrite,
            };

            var response = await client.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }


        /// <summary>
        /// Retrieves the ACL associated with the Amazon S3 bucket name in the
        /// bucketName parameter.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">The Amazon S3 bucket for which we want to
get the
        /// ACL list.</param>
        /// <returns>Returns an S3AccessControlList returned from the call to
        /// GetACLAsync.</returns>
        public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
client, string bucketName)
        {
            GetACLResponse response = await client.GetACLAsync(new GetACLRequest
            {
                BucketName = bucketName,
            });

            return response.AccessControlList;
        }



        /// <summary>
        /// Adds a new ACL to an existing object in the Amazon S3 bucket.
        /// </summary>
```

```
        /// <param name="client">The initialized client object used to call
        /// PutBucketAsync.</param>
        /// <param name="bucketName">A string representing the name of the Amazon
 S3
        /// bucket containing the object to which we want to apply a new ACL.</
param>
        /// <param name="keyName">A string representing the name of the object
        /// to which we want to apply the new ACL.</param>
        /// <param name="emailAddress">The email address of the person to whom
        /// we will be applying to whom access will be granted.</param>
        public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
 string bucketName, string keyName, string emailAddress)
        {
            // Retrieve the ACL for an object.
            GetACLResponse aclResponse = await client.GetACLAsync(new
 GetACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
            });

            S3AccessControlList acl = aclResponse.AccessControlList;

            // Retrieve the owner.
            Owner owner = acl.Owner;

            // Clear existing grants.
            acl.Grants.Clear();

            // Add a grant to reset the owner's full permission
            // (the previous clear statement removed all permissions).
            var fullControlGrant = new S3Grant
            {
                Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
            };
            acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

            // Specify email to identify grantee for granting permissions.
            var grantUsingEmail = new S3Grant
            {
                Grantee = new S3Grantee { EmailAddress = emailAddress },
                Permission = S3Permission.WRITE_ACP,
            };
```

```
            // Specify log delivery group as grantee.
            var grantLogDeliveryGroup = new S3Grant
            {
                Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
                Permission = S3Permission.WRITE,
            };

            // Create a new ACL.
            var newAcl = new S3AccessControlList
            {
                Grants = new List<S3Grant> { grantUsingEmail,
 grantLogDeliveryGroup },
                Owner = owner,
            };

            // Set the new ACL. We're throwing away the response here.
            _ = await client.PutACLAsync(new PutACLRequest
            {
                BucketName = bucketName,
                Key = keyName,
                AccessControlList = newAcl,
            });
        }

    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的以下主题。
  - GetBucketAcl
  - GetObjectAcl
  - PutBucketAcl
  - PutObjectAcl

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

通过 AWS SDK 使用 Lambda 函数批量管理版本控制的 Amazon S3 对象

以下代码示例显示了如何使用 Lambda 函数批量管理版本控制的 S3 对象。

Python

SDK for Python (Boto3)

显示如何通过创建调用 AWS Lambda 函数来执行处理的任务来批量操作 Amazon Simple Storage Service (Amazon S3) 版本控制对象。此示例将创建了一个启用版本控制的桶，上传 Lewis Carroll 所写的诗歌《You Are Old, Father William》中的诗节，并使用 Amazon S3 批处理任务以各种方式调整这首诗。

了解如何：

- 创建对版本控制对象运行的 Lambda 函数。
- 创建要更新的对象清单。
- 创建调用 Lambda 函数来更新对象的批处理任务。
- 删除 Lambda 函数。
- 清空并删除版本控制的桶。

此示例最好在 GitHub 上查看。有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Amazon S3

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 解析 Amazon S3 URI

以下代码示例显示如何解析 Amazon S3 URI 以提取诸如桶名称和对象密钥等重要组件。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用 S3Uri 类解析 Amazon S3 URI。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

    /**
     *
     * @param s3Client    - An S3Client through which you acquire an S3Uri
 instance.
     * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate
 S3Uri
     *                    capabilities.
     */
    public static void parseS3UriExample(S3Client s3Client, String s3ObjectUrl) {
        logger.info(s3ObjectUrl);
        // Console output:
        // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

        // Create an S3Utilities object using the configuration of the s3Client.
        S3Utilities s3Utilities = s3Client.utilities();

        // From a String URL create a URI object to pass to the parseUri()
 method.
        URI uri = URI.create(s3ObjectUrl);
        S3Uri s3Uri = s3Utilities.parseUri(uri);

        // If the URI contains no value for the Region, bucket or key, the SDK
 returns
        // an empty Optional.
        // The SDK returns decoded URI values.

        Region region = s3Uri.region().orElse(null);
        log("region", region);
        // Console output: 'region: us-west-1'.
```

```
        String bucket = s3Uri.bucket().orElse(null);
        log("bucket", bucket);
        // Console output: 'bucket: myBucket'.

        String key = s3Uri.key().orElse(null);
        log("key", key);
        // Console output: 'key: resources/doc.txt'.

        Boolean isPathStyle = s3Uri.isPathStyle();
        log("isPathStyle", isPathStyle);
        // Console output: 'isPathStyle: true'.

        // If the URI contains no query parameters, the SDK returns an empty map.
        Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
        log("rawQueryParameters", queryParams);
        // Console output: 'rawQueryParameters: {versionId=[abc123],
 partNumber=[77,
        // 88]}'.

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
 s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

        String partNumber =
 s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
 s3Uri.firstMatchingRawQueryParameters("partNumber");
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
 must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
```

```
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
 the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 执行 Amazon S3 对象的分段复制

以下代码示例显示如何执行 Amazon S3 对象的分段复制。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
```

```
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUapiCopyObj
{
    private const string SourceBucket = "doc-example-bucket1";
    private const string TargetBucket = "doc-example-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPUCopyObjectAsync(s3Client);
    }

    /// <summary>
    /// This method uses the passed client object to perform a multipart
    /// copy operation.
    /// </summary>
    /// <param name="client">An Amazon S3 client object that will be used
    /// to perform the copy.</param>
    public static async Task MPUCopyObjectAsync(AmazonS3Client client)
    {
        // Create a list to store the copy part responses.
        var copyResponses = new List<CopyPartResponse>();

        // Setup information required to initiate the multipart upload.
        var initiateRequest = new InitiateMultipartUploadRequest
        {
            BucketName = TargetBucket,
            Key = TargetObjectKey,
        };

        // Initiate the upload.
        InitiateMultipartUploadResponse initResponse =
```

```
                await client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            string uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                var metadataRequest = new GetObjectMetadataRequest
                {
                    BucketName = SourceBucket,
                    Key = SourceObjectKey,
                };

                GetObjectMetadataResponse metadataResponse =
                    await client.GetObjectMetadataAsync(metadataRequest);
                var objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    var copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = TargetBucket,
                        DestinationKey = TargetObjectKey,
                        SourceBucket = SourceBucket,
                        SourceKey = SourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i,
                    };

                    copyResponses.Add(await client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
```

```
                var completeRequest = new CompleteMultipartUploadRequest
                {
                    BucketName = TargetBucket,
                    Key = TargetObjectKey,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error encountered on server.
Message:'{e.Message}' when writing an object");
            }
            catch (Exception e)
            {
                Console.WriteLine($"Unknown encountered on server.
Message:'{e.Message}' when writing an object");
            }
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的以下主题。
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - GetObjectMetadata
  - UploadPartCopy

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 执行 Amazon S3 对象的分段上传

以下代码示例显示如何执行 Amazon S3 对象的分段上传。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

代码示例使用以下导入。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

在基于 AWS CRT 的 S3 客户端上使用 S3 Transfer Manager，以在内容大小超过阈值时透明地执行分段上传。默认阈值大小为 8 MB。

```
    public void multipartUploadWithTransferManager(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

使用 S3Client API 执行分段上传。

```
    public void multipartUploadWithS3Client(String filePath) {

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key));
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
```

```
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

使用启用分段支持的 S3AsyncClient API 执行分段上传。

```
    public void multipartUploadWithS3AsyncClient(String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .multipartEnabled(true)
                .build();
```

```
        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
                        .bucket(bucketName)
                        .key(key),
                Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using
S3AsyncClient.");
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的以下主题。
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 接收和处理 Amazon S3 事件通知。

以下代码示例显示了如何以面向对象的方式处理 S3 事件通知。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

此示例显示了如何使用 Amazon SQS 处理 S3 通知事件。

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload
and logs them. It uses
```

```
    * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
    * the JSON payload and access the messages in an object-oriented way.
    *
    * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
    * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
 API</a>.
    * <p>
    * To use S3 event notification serialization/deserialization to objects, add
 the following
    * dependency to your Maven pom.xml file.
    * <dependency>
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
 Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
 process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
 through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String
 queueArn) {
       try {
           // Configure the bucket to send Object Created and Object Tagging
 notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
 Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
```

```
            // Wait for event notifications to propagate.
            Thread.sleep(Duration.ofSeconds(5).toMillis());

            boolean didReceiveMessages = true;
            while (didReceiveMessages) {
                // Display the number of messages that are available in the
queue.
                sqsClient.getQueueAttributes(b -> b
                                .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                        ).thenAccept(attributeResponse ->
                                logger.info("Approximate number of messages in
the queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                        .join();

                // Receive the messages.
                ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                        .queueUrl(queueUrl)
                ).get();
                logger.info("Count of received messages: {}",
response.messages().size());
                didReceiveMessages = !response.messages().isEmpty();

                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification
object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
```

```
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}",
eventName, key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {

sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的以下主题。
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 保存 EXIF 和其他图像信息

以下代码示例展示了如何：

- 从 JPG、JPEG 或 PNG 文件中获取 EXIF 信息。
- 将图像文件上传到 Amazon S3 存储桶。
- 使用 Amazon Rekognition 识别文件中的三个主要属性（标签）。
- 将 EXIF 和标签信息添加到该区域的 Amazon DynamoDB 表中。

Rust

适用于 Rust 的 SDK

从 JPG、JPEG 或 PNG 文件中获取 EXIF 信息，将图像文件上传到 Amazon S3 存储桶，使用 Amazon Rekognition 识别文件中的三个主要属性（Amazon Rekognition 中的标签），然后将 EXIF 和标签信息添加到该区域的 Amazon DynamoDB 表中。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- DynamoDB
- Amazon Rekognition
- Amazon S3

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 向 Amazon EventBridge 发送 S3 事件通知

以下代码示例显示了如何启用存储桶来将 S3 事件通知发送到 EventBridge，并将通知路由到 Amazon SNS 主题和 Amazon SQS 队列。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
```

```
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的以下主题。
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

跟踪使用 AWS SDK 执行的 Amazon S3 对象上传或下载操作

下面的代码示例展示了如何跟踪 Amazon S3 对象的上传或下载。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

跟踪文件上传进度。

```
   public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                               String key, URI filePathURI) {
       UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
               .putObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .source(Paths.get(filePathURI))
               .build();

       FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

       fileUpload.completionFuture().join();
       /*
           The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
           You can also implement the interface to provide your own logic.

           Configure log4J2 with settings such as the following.
               <Configuration status="WARN">
                   <Appenders>
                       <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                           <PatternLayout pattern="%m%n"/>
                       </Console>
                   </Appenders>
```

```
                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file upload.
                Transfer initiated...
                |                    | 0.0%
                |====                | 21.1%
                |============        | 60.5%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

跟踪文件下载进度。

```
    public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                                  String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())  // Add
listener.
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        /*
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.
```

```
            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3
MB file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

- 有关 API 详细信息，请参阅 AWS SDK for Java 2.x API 参考中的以下主题。
  - GetObject
  - PutObject

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 S3 对象 Lambda 转换应用程序的数据

下面的代码示例显示如何使用 S3 对象 Lambda 转换应用程序的数据。

.NET

AWS SDK for .NET

显示如何将自定义代码添加到标准 S3 GET 请求中，来修改从 S3 检索的请求对象，从而使该对象适合发出请求的客户端或应用程序的需要。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

- Lambda
- Amazon S3

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 进行单元测试和集成测试的示例方法

以下代码示例显示了使用 AWS SDK 编写单元测试和集成测试时的最佳实践方法示例。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

Cargo.toml 用于测试示例。

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"
```

```
[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-
credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "~4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

[[bin]]
name = "main"
path = "src/main.rs"
```

使用 automock 和服务包装器的单元测试示例。

```
use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}

#[cfg_attr(test, automock)]
```

```
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
```

```
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
        // Create the Mock instance with two pages of objects now
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
```

```
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
                    .set_next_continuation_token(Some("next".to_string()))
                    .build())
            });
        mock.expect_list_objects()
            .with(
                eq("test-bucket"),
                eq("test-prefix"),
                eq(Some("next".to_string())),
            )
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(3).build(),
                        s3::types::Object::builder().size(9).build(),
                    ]))
                    .build())
            });

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
    }
}
```

使用 StaticReplayClient 的集成测试示例。

```
use aws_sdk_s3 as s3;

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
```

```
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}

#[cfg(test)]
mod test {
    use super::*;
```

```
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
 StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        // Verify we got the correct total size back
        assert_eq!(7, size);
        replay_client.assert_requests_match(&[]);
    }

    #[tokio::test]
    async fn test_multiple_pages() {
```

```
        let page_1 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
                    .unwrap(),
            );
        let page_2 = ReplayEvent::new(
                http::Request::builder()
                    .method("GET")
                    .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
                    .body(SdkBody::empty())
                    .unwrap(),
                http::Response::builder()
                    .status(200)
                    .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))
                    .unwrap(),
            );
        let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()
                .behavior_version(BehaviorVersion::latest())
                .credentials_provider(make_s3_test_credentials())
                .region(s3::config::Region::new("us-east-1"))
                .http_client(replay_client.clone())
                .build(),
        );

        // Run the code we want to test with it
        let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
```

```
        replay_client.assert_requests_match(&[]);
    }
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

以递归方式将本地目录上传到 Amazon Simple Storage Service（Amazon S3）桶

以下代码示例显示如何以递归方式将本地目录上传到 Amazon Simple Storage Service（Amazon S3）桶。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用 S3TransferManager 以上传本地目录。查看完整文件并进行测试。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;
```

```
    public Integer uploadDirectory(S3TransferManager transferManager,
            URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
                .source(Paths.get(sourceDirectory))
                .bucket(bucketName)
                .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 UploadDirectory。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

借助 AWS SDK 向 Amazon S3 上传大文件或从 Amazon S3 下载大文件

下面的代码示例说明如何向 Amazon S3 上传大文件或从 Amazon S3 下载大文件。

有关更多信息，请参阅使用分段上传操作上传对象。

.NET

AWS SDK for .NET

> **Note**
>
> 在 GitHub 上查看更多内容。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

调用使用 Amazon S3 TransferUtility 将文件传入和传出 S3 存储桶的函数。

```
global using System.Text;
global using Amazon.S3;
```

```
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;



// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
 $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
 bucketName, fileToUpload, localPath);
if (success)
```

```
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
 {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
 bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
 {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
 keyName, localPath);
if (success)
```

```
{
    Console.WriteLine("$Successfully downloaded the file, {keyName} from
 {bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
 bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
 {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);
```

```
    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
 Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
 an\n\t  S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
 directory.");
    Console.WriteLine($"\n{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
 textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
```

```
    }
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
 s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}
```

上传单个文件。

```
        /// <summary>
        /// Uploads a single file from the local computer to an S3 bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the file
        /// will be stored.</param>
        /// <param name="fileName">The name of the file to upload.</param>
```

```
        /// <param name="localPath">The local path where the file is stored.</
param>
        /// <returns>A boolean value indicating the success of the action.</
returns>
        public static async Task<bool> UploadSingleFileAsync(
            TransferUtility transferUtil,
            string bucketName,
            string fileName,
            string localPath)
        {
            if (File.Exists($"{localPath}\\{fileName}"))
            {
                try
                {
                    await transferUtil.UploadAsync(new
 TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        Key = fileName,
                        FilePath = $"{localPath}\\{fileName}",
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Could not upload {fileName} from
 {localPath} because:");
                    Console.WriteLine(s3Ex.Message);
                    return false;
                }
            }
            else
            {
                Console.WriteLine($"{fileName} does not exist in {localPath}");
                return false;
            }
        }
```

上传整个本地目录。

```
        /// <summary>
        /// Uploads all the files in a local directory to a directory in an S3
        /// bucket.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket where the files
        /// will be stored.</param>
        /// <param name="keyPrefix">The key prefix is the S3 directory where
        /// the files will be stored.</param>
        /// <param name="localPath">The local directory that contains the files
        /// to be uploaded.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> UploadFullDirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string keyPrefix,
            string localPath)
        {
            if (Directory.Exists(localPath))
            {
                try
                {
                    await transferUtil.UploadDirectoryAsync(new
 TransferUtilityUploadDirectoryRequest
                    {
                        BucketName = bucketName,
                        KeyPrefix = keyPrefix,
                        Directory = localPath,
                    });

                    return true;
                }
                catch (AmazonS3Exception s3Ex)
                {
                    Console.WriteLine($"Can't upload the contents of {localPath}
 because:");
                    Console.WriteLine(s3Ex?.Message);
                    return false;
                }
            }
            else
            {
```

```
                Console.WriteLine($"The directory {localPath} does not exist.");
                return false;
            }
        }
```

下载单个文件。

```
        /// <summary>
        /// Download a single file from an S3 bucket to the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The name of the S3 bucket containing the
        /// file to download.</param>
        /// <param name="keyName">The name of the file to download.</param>
        /// <param name="localPath">The path on the local computer where the
        /// downloaded file will be saved.</param>
        /// <returns>A Boolean value indicating the results of the action.</
returns>
        public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
            string bucketName,
            string keyName,
            string localPath)
        {
            await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = $"{localPath}\\{keyName}",
            });

            return (File.Exists($"{localPath}\\{keyName}"));
        }
```

下载 S3 存储桶的内容。

```
        /// <summary>
        /// Downloads the contents of a directory in an S3 bucket to a
        /// directory on the local computer.
        /// </summary>
        /// <param name="transferUtil">The transfer initialized TransferUtility
        /// object.</param>
        /// <param name="bucketName">The bucket containing the files to
 download.</param>
        /// <param name="s3Path">The S3 directory where the files are located.</
param>
        /// <param name="localPath">The local path to which the files will be
        /// saved.</param>
        /// <returns>A Boolean value representing the success of the action.</
returns>
        public static async Task<bool> DownloadS3DirectoryAsync(
            TransferUtility transferUtil,
            string bucketName,
            string s3Path,
            string localPath)
        {
            int fileCount = 0;

            // If the directory doesn't exist, it will be created.
            if (Directory.Exists(s3Path))
            {
                var files = Directory.GetFiles(localPath);
                fileCount = files.Length;
            }

            await transferUtil.DownloadDirectoryAsync(new
 TransferUtilityDownloadDirectoryRequest
            {
                BucketName = bucketName,
                LocalDirectory = localPath,
                S3Directory = s3Path,
            });

            if (Directory.Exists(localPath))
            {
                var files = Directory.GetFiles(localPath);
                if (files.Length > fileCount)
                {
                    return true;
                }
```

```
            // No change in the number of files. Assume
            // the download failed.
            return false;
        }

        // The local directory doesn't exist. No files
        // were downloaded.
        return false;
    }
```

使用 TransferUtility 跟踪上传的进度。

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "doc-example-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }
```

```
        /// <summary>
        /// Starts an Amazon S3 multipart upload and assigns an event handler to
        /// track the progress of the upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client object used to
        /// perform the multipart upload.</param>
        /// <param name="bucketName">The name of the bucket to which to upload
        /// the file.</param>
        /// <param name="filePath">The path, including the file name of the
        /// file to be uploaded to the Amazon S3 bucket.</param>
        /// <param name="keyName">The file name to be used in the
        /// destination Amazon S3 bucket.</param>
        public static async Task TrackMPUAsync(
            IAmazonS3 client,
            string bucketName,
            string filePath,
            string keyName)
        {
            try
            {
                var fileTransferUtility = new TransferUtility(client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName,
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>(
                        UploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error:: {ex.Message}");
            }
        }
```

```
        /// <summary>
        /// Event handler to check the progress of the multipart upload.
        /// </summary>
        /// <param name="sender">The object that raised the event.</param>
        /// <param name="e">The object that contains multipart upload
        /// information.</param>
        public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
        }
    }
```

上传经过加密的对象。

```
    using System;
    using System.Collections.Generic;
    using System.IO;
    using System.Security.Cryptography;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
    /// perform a multipart upload to an Amazon S3 bucket.
    /// </summary>
    public class SSECLowLevelMPUcopyObject
    {
        public static async Task Main()
        {
            string existingBucketName = "doc-example-bucket";
            string sourceKeyName = "sample_file.txt";
            string targetKeyName = "sample_file_copy.txt";
            string filePath = $"sample\\{targetKeyName}";

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
```

```
            // For example: RegionEndpoint.USEast1.
            IAmazonS3 client = new AmazonS3Client();

            // Create the encryption key.
            var base64Key = CreateEncryptionKey();

            await CreateSampleObjUsingClientEncryptionKeyAsync(
                client,
                existingBucketName,
                sourceKeyName,
                filePath,
                base64Key);
        }

        /// <summary>
        /// Creates the encryption key to use with the multipart upload.
        /// </summary>
        /// <returns>A string containing the base64-encoded key for encrypting
        /// the multipart upload.</returns>
        public static string CreateEncryptionKey()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);
            return base64Key;
        }

        /// <summary>
        /// Creates and uploads an object using a multipart upload.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object used to
        /// initialize and perform the multipart upload.</param>
        /// <param name="existingBucketName">The name of the bucket to which
        /// the object will be uploaded.</param>
        /// <param name="sourceKeyName">The source object name.</param>
        /// <param name="filePath">The location of the source object.</param>
        /// <param name="base64Key">The encryption key to use with the upload.</
param>
        public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
            IAmazonS3 client,
            string existingBucketName,
            string sourceKeyName,
            string filePath,
```

```
            string base64Key)
        {
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
               await client.InitiateMultipartUploadAsync(initiateRequest);

            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key,
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));
```

```
                filePosition += partSize;
            }

            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                UploadId = initResponse.UploadId,
            };
            completeRequest.AddPartETags(uploadResponses);

            CompleteMultipartUploadResponse completeUploadResponse =
                await client.CompleteMultipartUploadAsync(completeRequest);
        }
        catch (Exception exception)
        {
            Console.WriteLine($"Exception occurred: {exception.Message}");

            // If there was an error, abort the multipart upload.
            AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                UploadId = initResponse.UploadId,
            };

            await client.AbortMultipartUploadAsync(abortMPURequest);
        }
    }
}
```

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用上传管理器上传大型对象，以将数据分成多个分段，然后同时上传。

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
 actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
 bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}



// UploadLargeObject uses an upload manager to upload data to an object in a
 bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(bucketName string, objectKey string,
 largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
```

```
   bucketName, objectKey, err)
 }

 return err
}
```

使用下载管理器下载大型对象，以分段获取数据并同时下载它们。

```
// DownloadLargeObject uses a download manager to download an object from a
 bucket.
// The download manager gets the data in parts and writes them to a buffer until
 all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(bucketName string, objectKey
 string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
 {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(context.TODO(), buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
 return buffer.Bytes(), err
}
```

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

调用使用 S3TransferManager 将文件传入和传出 S3 存储桶的函数。

```
   public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
           URI destinationPathURI, String bucketName) {
       DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
               .destination(Paths.get(destinationPathURI))
               .bucket(bucketName)
               .build());
       CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

       completedDirectoryDownload.failedTransfers()
               .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
       return completedDirectoryDownload.failedTransfers().size();
   }
```

上传整个本地目录。

```
   public Integer uploadDirectory(S3TransferManager transferManager,
           URI sourceDirectory, String bucketName) {
       DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
               .source(Paths.get(sourceDirectory))
               .bucket(bucketName)
               .build());

       CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
       completedDirectoryUpload.failedTransfers()
```

```
                .forEach(fail -> logger.warn("Object [{}] failed to transfer",
 fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

上传单个文件。

```
    public String uploadFile(S3TransferManager transferManager, String
bucketName,
                             String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

上传大文件。

```
import {
  CreateMultipartUploadCommand,
  UploadPartCommand,
  CompleteMultipartUploadCommand,
  AbortMultipartUploadCommand,
  S3Client,
```

```
} from "@aws-sdk/client-s3";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

export const main = async () => {
  const s3Client = new S3Client({});
  const bucketName = "test-bucket";
  const key = "multipart.txt";
  const str = createString();
  const buffer = Buffer.from(str, "utf8");

  let uploadId;

  try {
    const multipartUpload = await s3Client.send(
      new CreateMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );

    uploadId = multipartUpload.UploadId;

    const uploadPromises = [];
    // Multipart uploads require a minimum size of 5 MB per part.
    const partSize = Math.ceil(buffer.length / 5);

    // Upload each part.
    for (let i = 0; i < 5; i++) {
      const start = i * partSize;
      const end = start + partSize;
      uploadPromises.push(
        s3Client
          .send(
            new UploadPartCommand({
              Bucket: bucketName,
              Key: key,
              UploadId: uploadId,
              Body: buffer.subarray(start, end),
              PartNumber: i + 1,
```

```
            }),
          )
          .then((d) => {
            console.log("Part", i + 1, "uploaded");
            return d;
          }),
      );
    }

    const uploadResults = await Promise.all(uploadPromises);

    return await s3Client.send(
      new CompleteMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
        MultipartUpload: {
          Parts: uploadResults.map(({ ETag }, i) => ({
            ETag,
            PartNumber: i + 1,
          })),
        },
      }),
    );

    // Verify the output by downloading the file from the Amazon Simple Storage
 Service (Amazon S3) console.
    // Because the output is a 25 MB string, text editors might struggle to open
 the file.
  } catch (err) {
    console.error(err);

    if (uploadId) {
      const abortCommand = new AbortMultipartUploadCommand({
        Bucket: bucketName,
        Key: key,
        UploadId: uploadId,
      });

      await s3Client.send(abortCommand);
    }
  }
};
```

下载大文件。

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream } from "fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: parseInt(start),
    end: parseInt(end),
    length: parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

// When downloading a large file, you might want to break it down into
// smaller pieces. Amazon S3 accepts a Range header to specify the start
// and end of the byte range to be downloaded.
const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));
```

```
  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    console.log(`Downloading bytes ${nextRange.start} to ${nextRange.end}`);

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

export const main = async () => {
  await downloadInChunks({
    bucket: "my-cool-bucket",
    key: "my-cool-object.txt",
  });
};
```

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用多种可用的传输管理器设置创建传输文件的功能。在文件传输过程中，使用回调类写入回调进度。

```
import sys
```

```
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
```

```
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
```

```
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
```

```
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
 sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

演示传输管理器的功能并报告结果。

```
import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
```

```
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")


class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
 files? "
            )
```

```
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? "
            )
            try:
                self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
            except ParamValidationError as err:
                print(err)
                self.demo_bucket = None
            except ClientError as err:
                print(err)
                print(
                    f"Either {self.demo_bucket} doesn't exist or you don't "
                    f"have access to it."
                )
                self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(
```

```
                TransferConfig, CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Uploading", local_file_path, object_key, **upload_args
            )
            start_time = time.perf_counter()
            thread_info = upload_func(
                local_file_path,
                self.demo_bucket,
                object_key,
                self.file_size_mb,
                **upload_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

            file_transfer.TransferConfig = self._config_wrapper(
                TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
            )
            self._report_transfer_params(
                "Downloading", object_key, download_file_path, **download_args
            )
            start_time = time.perf_counter()
            thread_info = download_func(
                self.demo_bucket,
                object_key,
                download_file_path,
                self.file_size_mb,
                **download_args,
            )
            end_time = time.perf_counter()
            self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print("-" * self._terminal_width)
```

```
        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)

            if self.demo_bucket:
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:

self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

    def _setup_platform_specific(self):
        """Set up platform-specific command used to create a large file."""
        if platform.system() == "Windows":
            self._create_file_cmd = "fsutil file createnew {} {}"
            self._size_multiplier = MB
        elif platform.system() == "Linux" or platform.system() == "Darwin":
            self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
            self._size_multiplier = 1
        else:
            raise EnvironmentError(
                f"Demo of platform {platform.system()} isn't supported."
            )

    def _create_demo_file(self):
        """
        Create a file in the demo folder specified by the user. Store the local
        path, object name, and download path for later cleanup.

        Only the local file is created by this method. The Amazon S3 object and
        download file are created later during the demonstration.
```

```
        Returns:
        A tuple that contains the local file path, object name, and download
        file path.
        """
        file_name_template = "TestFile{}-{}.demo"
        local_suffix = "local"
        object_suffix = "s3object"
        download_suffix = "downloaded"
        file_tag = len(self._chore_list) + 1

        local_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag, local_suffix)
        )

        s3_object_key = file_name_template.format(file_tag, object_suffix)

        downloaded_file_path = os.path.join(
            self.demo_folder, file_name_template.format(file_tag,
download_suffix)
        )

        filled_cmd = self._create_file_cmd.format(
            local_file_path, self.file_size_mb * self._size_multiplier
        )

        print(
            f"Creating file of size {self.file_size_mb} MB "
            f"in {self.demo_folder} by running:"
        )
        print(f"{'':4}{filled_cmd}")
        os.system(filled_cmd)

        chore = (local_file_path, s3_object_key, downloaded_file_path)
        self._chore_list.append(chore)
        return chore

    def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
        """Report configuration and extra arguments used for a file transfer."""
        print("-" * self._terminal_width)
        print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
        if kwargs:
            print("With extra args:")
            for arg, value in kwargs.items():
                print(f'{"":4}{arg:<20}: {value}')
```

```
    @staticmethod
    def ask_user(question):
        """
        Ask the user a yes or no question.

        Returns:
        True when the user answers 'y' or 'Y'; otherwise, False.
        """
        answer = input(f"{question} (y/n) ")
        return answer.lower() == "y"

    @staticmethod
    def _config_wrapper(func, config_attrs):
        def wrapper(*args, **kwargs):
            config = func(*args, **kwargs)
            print("With configuration:")
            for attr in config_attrs:
                print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
            return config

        return wrapper

    @staticmethod
    def _report_transfer_result(thread_info, elapsed):
        """Report the result of a transfer, including per-thread data."""
        print(f"\nUsed {len(thread_info)} threads.")
        for ident, byte_count in thread_info.items():
            print(f"{'':4}Thread {ident} copied {byte_count} bytes.")
        print(f"Your transfer took {elapsed:.2f} seconds.")


def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
```

```
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )
```

```
    # Download without specifying an encryption key to show that the
    # encryption key must be included to download an encrypted object.
    if demo_manager.ask_user(
        "Do you want to try to download the encrypted "
        "object without sending the required key?"
    ):
        try:
            _, object_key, download_file_path = demo_manager.last_name_set()
            file_transfer.download_with_default_configuration(
                demo_manager.demo_bucket,
                object_key,
                download_file_path,
                demo_manager.file_size_mb,
            )
        except ClientError as err:
            print(
                "Got expected error when trying to download an encrypted "
                "object without specifying encryption info:"
            )
            print(f"{'':4}{err}")

    # Remove all created and downloaded files, remove all objects from
    # S3 storage.
    if demo_manager.ask_user(
        "Demonstration complete. Do you want to remove local files " "and S3
 objects?"
    ):
        demo_manager.cleanup()


if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
 get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_service::error::Error;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}
```

```
async fn run_example() -> Result<(), Error> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;

    let key = "sample.txt".to_string();
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await
        .unwrap();
    let upload_id = multipart_upload_res.upload_id().unwrap();

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
        .len();

    let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
    let mut size_of_last_chunk = file_size % CHUNK_SIZE;
```

```
    if size_of_last_chunk == 0 {
        size_of_last_chunk = CHUNK_SIZE;
        chunk_count -= 1;
    }

    if file_size == 0 {
        panic!("Bad file size.");
    }
    if chunk_count > MAX_CHUNKS {
        panic!("Too many chunks! Try increasing your chunk size.")
    }

    let mut upload_parts: Vec<CompletedPart> = Vec::new();

    for chunk_index in 0..chunk_count {
        let this_chunk = if chunk_count - 1 == chunk_index {
            size_of_last_chunk
        } else {
            CHUNK_SIZE
        };
        let stream = ByteStream::read_from()
            .path(path)
            .offset(chunk_index * CHUNK_SIZE)
            .length(Length::Exact(this_chunk))
            .build()
            .await
            .unwrap();
        //Chunk index needs to start at 0, but part numbers start at 1.
        let part_number = (chunk_index as i32) + 1;
        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;
        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
```

```
    }
    let completed_multipart_upload: CompletedMultipartUpload =
 CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

    let _complete_multipart_upload_res = client
        .complete_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .multipart_upload(completed_multipart_upload)
        .upload_id(upload_id)
        .send()
        .await
        .unwrap();

    let data: GetObjectOutput = s3_service::download_object(&client,
 &bucket_name, &key).await?;
    let data_length: u64 = data
        .content_length()
        .unwrap_or_default()
        .try_into()
        .unwrap();
    if file.metadata().unwrap().len() == data_length {
        println!("Data lengths match.");
    } else {
        println!("The data was not the same size!");
    }

    s3_service::delete_objects(&client, &bucket_name)
        .await
        .expect("Error emptying bucket.");
    s3_service::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");

    Ok(())
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 将未知大小的流上传到 Amazon S3 对象

下面的代码示例展示了如何将未知大小的流上传到 Amazon S3 对象。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用 AWS 基于 CRT 的 S3 客户端。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

    /**
     * @param s33CrtAsyncClient - To upload content from a stream of unknown
 size, use the AWS CRT-based S3 client. For more information, see
     *                          https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/crt-based-s3-client.html.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
 Returns metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStream(S3AsyncClient s33CrtAsyncClient,
 String bucketName, String key) {
```

```
        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null'
 indicates a stream will be provided later.

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        return response;
    }
}
```

使用 Amazon S3 Transfer Manager。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.util.UUID;

    /**
     * @param transferManager - To upload content from a stream of unknown size,
 use the S3TransferManager based on the AWS CRT-based S3 client.
```

```
     *                       For more information, see https://
docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
 bucketName, String key) {

        BlockingInputStreamAsyncRequestBody body =
                AsyncRequestBody.forBlockingInputStream(null); // 'null'
 indicates a stream will be provided later.

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());

        // Provide the stream of data to be uploaded.
        body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

        return upload.completionFuture().join();
    }
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS 开发工具包，对 Amazon S3 对象使用校验和

以下代码示例显示如何对 Amazon S3 对象使用校验和。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

代码示例使用以下导入的子集。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
```

```
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

在构建 PutObjectRequest 时为 putObject 方法指定校验和算法。

```
    public void putObjectWithChecksum() {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
                RequestBody.fromString("This is a test"));
    }
```

在构建 GetObjectRequest 时为 getObject 方法验证校验和。

```
    public GetObjectResponse getObjectWithChecksum() {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
                .response();
    }
```

在构建 PutObjectRequest 时为 putObject 方法预先计算校验和。

```
    public void putObjectWithPrecalculatedChecksum(String filePath) {
        String checksum = calculateChecksum(filePath, "SHA-256");

        s3Client.putObject((b -> b
                .bucket(bucketName)
                .key(key)
                .checksumSHA256(checksum)),
                RequestBody.fromFile(Paths.get(filePath)));
    }
```

在基于 AWS CRT 的 S3 客户端上使用 S3 Transfer Manager，以在内容大小超过阈值时透明地执行分段上传。默认阈值大小为 8 MB。

您可以为要使用的开发工具包指定校验和算法。默认情况下，开发工具包使用 CRC32 算法。

```
    public void multipartUploadWithChecksumTm(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
                .putObjectRequest(b -> b
                        .bucket(bucketName)
                        .key(key)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA1))
                .source(Paths.get(filePath))
                .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

使用 S3Client API 或 S3AsyncClient API 执行分段上传。如果指定其他校验和，则您必须指定在启动上传时使用的算法。您还必须为每个分段请求指定算法，并提供每个分段在上传后计算得出的校验和。

```
    public void multipartUploadWithChecksumS3Client(String filePath) {
        ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(algorithm)); // Checksum specified on
initiation.
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
```

```
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .checksumAlgorithm(algorithm) // Checksum specified on
each part.
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .checksumCRC32(partResponse.checksumCRC32()) // Provide
the calculated checksum.
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException e) {
            System.err.println(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的以下主题。

- CompleteMultipartUpload
- CreateMultipartUpload
- UploadPart

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

通过 AWS SDK 使用 Amazon S3 对象完整性功能

以下代码示例显示了如何使用 S3 对象完整性功能。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

运行演示 Amazon S3 对象完整性功能的交互式场景。

```
//! Routine which runs the S3 object integrity workflow.
/*!
   \param clientConfig: Aws client configuration.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
        const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
 std::endl;
        return false;
    }
```

```
    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
std::endl;
        return false;
    }

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
    std::shared_ptr<Aws::S3::S3Client> client =
Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

    printAsterisksLine();
    std::cout << "Choose from one of the following checksum algorithms."
              << std::endl;

    for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
        std::cout << "  " << hashMethod << " - " <<
stringForHashMethod(hashMethod)
                  << std::endl;
    }

    HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
DEFAULT,
                                                          SHA256);


    gUseCalculatedChecksum = !askYesNoQuestion(
            "Let the SDK calculate the checksum for you? (y/n) ");

    printAsterisksLine();

    std::cout << "The workflow will now upload a file using PutObject."
              << std::endl;
    std::cout << "Object integrity will be verified using the "
              << stringForHashMethod(chosenHashMethod) << " algorithm."
              << std::endl;
    if (gUseCalculatedChecksum) {
```

```
        std::cout
                << "A checksum computed by this workflow will be used for object
integrity verification,"
                << std::endl;
        std::cout << "except for the TransferManager upload." << std::endl;
    } else {
        std::cout
                << "A checksum computed by the SDK will be used for object
integrity verification."
                << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();

    std::shared_ptr<Aws::IOStream> inputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Hasher hasher;
    HASH_METHOD putObjectHashMethod = chosenHashMethod;
    if (putObjectHashMethod == DEFAULT) {
        putObjectHashMethod = MD5; // MD5 is the default hash method for
PutObject.

        std::cout << "The default checksum algorithm for PutObject is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
    }

    // Demonstrate in code how the hash is computed.
    if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
        std::cerr << "Error calculating hash for file " << TEST_FILE <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
```

```
    }
    Aws::String key = stringForHashMethod(putObjectHashMethod);
    key += "_";
    key += TEST_FILE_KEY;
    Aws::String localHash = hasher.getBase64HashString();

    // Upload the object with PutObject
    if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                           inputData, chosenHashMethod == DEFAULT,
                           *client)) {
        std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
                            putObjectHashMethod, retrievedHash,
                            nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                  << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                         "PutObject upload", putObjectHashMethod);


    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32;  // The default hash method for the
TransferManager is CRC32.
```

```
        std::cout << "The default checksum algorithm for TransferManager is "
                  << stringForHashMethod(transferManagerHashMethod)
                  << std::endl;
    }

    // Upload the large file using the transfer manager.
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
chosenHashMethod == DEFAULT,
                                 client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::vector<Aws::String> retrievedTransferManagerPartHashes;
    Aws::String retrievedTransferManagerFinalHash;

    // Retrieve all the hashes for the TransferManager upload.
    if (!retrieveObjectHash(bucketName, key,
                            transferManagerHashMethod,
                            retrievedTransferManagerFinalHash,
                            &retrievedTransferManagerPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    AwsDoc::S3::Hasher locallyCalculatedFinalHash;
    std::vector<Aws::String> locallyCalculatedPartHashes;

    // Calculate the hashes locally to demonstrate how TransferManager hashes are
computed.
    if (!calculatePartHashesForFile(transferManagerHashMethod,
MULTI_PART_TEST_FILE,
                                    UPLOAD_BUFFER_SIZE,
                                    locallyCalculatedFinalHash,
                                    locallyCalculatedPartHashes)) {
        std::cerr << "Exiting because of an error in calculatePartHashesForFile."
<< std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
```

```
    verifyHashingResults(retrievedTransferManagerFinalHash,
                         locallyCalculatedFinalHash, "TransferManager upload",
                         transferManagerHashMethod,
                         retrievedTransferManagerPartHashes,
                         locallyCalculatedPartHashes);

    printAsterisksLine();

    key = "mp_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    multiPartUploadExplanations(key, chosenHashMethod);

    pressEnterToContinue();

    std::shared_ptr<Aws::IOStream> largeFileInputData =
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          MULTI_PART_TEST_FILE,
                                          std::ios_base::in |
                                          std::ios_base::binary);

    if (!largeFileInputData->good()) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
    if (multipartUploadHashMethod == DEFAULT) {
        multipartUploadHashMethod = MD5;  // The default hash method for
multipart uploads is MD5.

        std::cout << "The default checksum algorithm for multipart upload is "
                  << stringForHashMethod(putObjectHashMethod)
                  << std::endl;
    }

    AwsDoc::S3::Hasher hashData;
    std::vector<Aws::String> partHashes;

    if (!doMultipartUpload(bucketName, key,
                           multipartUploadHashMethod,
                           largeFileInputData, chosenHashMethod == DEFAULT,
```

```
                            hashData,
                            partHashes,
                            *client)) {
        std::cerr << "Exiting because of an error in doMultipartUpload." <<
 std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::cout << "Finished multipart upload of with hash method " <<
              stringForHashMethod(multipartUploadHashMethod) << std::endl;

    std::cout << "Now we will retrieve the checksums from the server." <<
 std::endl;

    retrievedHash.clear();
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
                            multipartUploadHashMethod,
                            retrievedHash, &retrievedPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
 multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                         multipartUploadHashMethod,
                         retrievedPartHashes, partHashes);

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
 workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
        std::cout << "The bucket " << bucketName << " was not deleted." <<
 std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object
 integrity hashing methods.
```

```
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashData: The hash value that will be associated with the uploaded
 object.
   \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
   \param body: The data content of the object being uploaded.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
 &key,
                                   const Aws::String &hashData,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &body,
                                   bool useDefaultHashMethod,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {

 request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
```

```
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
    request.SetBody(body);
    Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
    body->seekg(0, body->beg);
    if (outcome.IsSuccess()) {
        std::cout << "Object successfully uploaded." << std::endl;
    } else {
        std::cerr << "Error uploading object." <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}


// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
   \param bucket: The name of the S3 bucket where the object is stored.
   \param key: The unique identifier (key) of the object within the S3 bucket.
   \param hashMethod: The hashing algorithm used to calculate the hash value of
 the object.
   \param[out] hashData: The retrieved hash.
   \param[out] partHashes: The part hashes if available.
   \param client: The S3 client instance used to retrieve the object.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    Aws::String &hashData,
                                    std::vector<Aws::String> *partHashes,
                                    const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
```

```
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
 client.GetObjectAttributes(
                request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
 outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break;  // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break;  // MD5 is not supported.
#pragma clang diagnostic pop
                case AwsDoc::S3::SHA1:
                    hashData = result.GetChecksum().GetChecksumSHA1();
                    break;
                case AwsDoc::S3::SHA256:
                    hashData = result.GetChecksum().GetChecksumSHA256();
                    break;
                case AwsDoc::S3::CRC32:
                    hashData = result.GetChecksum().GetChecksumCRC32();
                    break;
                case AwsDoc::S3::CRC32C:
```

```
                    hashData = result.GetChecksum().GetChecksumCRC32C();
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        } else {
            std::cerr << "Error retrieving object checksum attributes." <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
```

```
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object
 parts." <<
                          outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }

    return true;
}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
 \param retrievedHash The hash value retrieved from the remote source.
 \param localHash The hash value calculated locally.
 \param uploadtype The type of upload (e.g., "multipart", "single-part").
 \param hashMethod The hashing method used (e.g., MD5, SHA-256).
 \param retrievedPartHashes (Optional) The list of hashes for the individual
 parts retrieved from the remote source.
 \param localPartHashes (Optional) The list of hashes for the individual parts
 calculated locally.
 */
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                      const Hasher &localHash,
                                      const Aws::String &uploadtype,
                                      HASH_METHOD hashMethod,
                                      const std::vector<Aws::String>
 &retrievedPartHashes,
                                      const std::vector<Aws::String>
 &localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash
 << std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
 retrieved."
                  << std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << "  Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
```

```
   if (hashMethod == MD5) {
       hashString = localHash.getHexHashString();
       if (!localPartHashes.empty()) {
           hashString += "-" + std::to_string(localPartHashes.size());
       }
   } else {
       hashString = localHash.getBase64HashString();
   }

   bool allMatch = true;
   if (hashString != retrievedHash) {
       std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
std::endl;
       std::cerr << "Local hash- '" << hashString << "'" << std::endl;
       std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
       allMatch = false;
   }

   if (hashMethod != MD5) {
       if (localPartHashes.size() != retrievedPartHashes.size()) {
           std::cerr << "For " << uploadtype << ", the number of part hashes do
not match" << std::endl;
           std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
"'"
                     << std::endl;
           std::cerr << "Remote number of hashes - '"
                     << retrievedPartHashes.size()
                     << "'" << std::endl;
       }

       for (int i = 0; i < localPartHashes.size(); ++i) {
           if (localPartHashes[i] != retrievedPartHashes[i]) {
               std::cerr << "For " << uploadtype << ", the part hashes do not
match for part " << i + 1
                         << "." << std::endl;
               std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                         << std::endl;
               std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                         << std::endl;
               allMatch = false;
           }
       }
   }
```

```
    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
 hashes all match!" << std::endl;
    }

}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &,
                                         const
 Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" <<
 std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                          const std::shared_ptr<const
 Aws::Transfer::TransferHandle> &handle) {
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
 std::endl;
    }
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's
 Transfer Manager.
/*!
   \param bucket: The name of the S3 bucket where the object will be uploaded.
   \param key: The unique identifier (key) for the object within the S3 bucket.
   \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
   \param useDefaultHashMethod: A flag indicating whether to use the default hash
 method or the one specified in the hashMethod parameter.
   \param client: The S3 client instance used to perform the upload operation.
   \return bool: Function succeeded.
*/
bool
AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
 &key,
                                    AwsDoc::S3::HASH_METHOD hashMethod,
                                    bool useDefaultHashMethod,
                                    const std::shared_ptr<Aws::S3::S3Client>
 &client) {
```

```
    std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
 Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
            "executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = client;
    transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
    if (!useDefaultHashMethod) {
        if (hashMethod == MD5) {
            transfer_config.computeContentMD5 = true;
        } else {
            transfer_config.checksumAlgorithm =
 getChecksumAlgorithmForHashMethod(
                    hashMethod);
        }
    }
    transfer_config.errorCallback = transferManagerErrorCallback;
    transfer_config.transferStatusUpdatedCallback =
 transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
 Aws::Transfer::TransferManager::Create(
            transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle =
 transfer_manager->UploadFile(MULTI_PART_TEST_FILE,

              bucket, key,

              "text/plain",

              Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
            uploadHandle->GetStatus() ==
 Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle-
>GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}
```

```
//! Routine which calculates the hash values for each part of a file being
 uploaded to an S3 bucket.
/*!
   \param hashMethod: The hashing algorithm to use when calculating the hash
 values.
   \param fileName: The path to the file for which the part hashes will be
 calculated.
   \param bufferSize: The size of the buffer to use when reading the file.
   \param[out] hashDataResult: The Hasher object that will store the concatenated
 hash value.
   \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
   \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                            const Aws::String &fileName,
                                            size_t bufferSize,
                                            AwsDoc::S3::Hasher &hashDataResult,
                                            std::vector<Aws::String> &partHashes)
 {
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;


    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
 static_cast<std::streamsize>(std::min(buffer.size(), objectSize -
 uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
 preallocatedStreamBuf(buffer.data(),

 bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
                Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                               &preallocatedStreamBuf);
        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
```

```
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
 hashBuffer.GetUnderlyingData(),
                               hashBuffer.GetUnderlyingData() +
 hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
 &key,
                                  Aws::S3::Model::ChecksumAlgorithm
 checksumAlgorithm,
                                  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
            client.CreateMultipartUpload(request);

    Aws::String uploadID;
```

```
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
 checksum algorithm,
                           ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
 &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
 &uploadID,
                                                         int partNumber,

 Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
 &calculatedHash,
                                                         const
 std::shared_ptr<Aws::IOStream> &body,
                                                         const Aws::S3::S3Client
 &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
```

```
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                      const Aws::String &key,
                                      const Aws::String &uploadID,
                                      const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
```

```
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
 AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

  const Aws::String &key,

  const Aws::String &uploadID,

  const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

  const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
```

```
            client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
 outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

//! Routine which performs a multi-part upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash
 value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default
 hash method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the
 concatenated hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
 for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream>
 &ioStream,
                                   bool useDefaultHashMethod,
                                   AwsDoc::S3::Hasher &hashDataResult,
                                   std::vector<Aws::String> &partHashes,
                                   const Aws::S3::S3Client &client) {
    // Get object size.
    ioStream->seekg(0, ioStream->end);
    size_t objectSize = ioStream->tellg();
    ioStream->seekg(0, ioStream->beg);

    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
 Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
```

```
        }
    }
    Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
client);
    if (uploadID.empty()) {
        return false;
    }

    std::vector<unsigned char> totalHashBuffer;
    bool uploadSucceeded = true;
    std::streamsize uploadedBytes = 0;
    int partNumber = 1;
    Aws::Vector<Aws::S3::Model::CompletedPart> parts;
    while (uploadedBytes < objectSize) {
        std::cout << "Uploading part " << partNumber << "." << std::endl;

        std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
        std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),

objectSize - uploadedBytes));
        ioStream->read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
                Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                               &preallocatedStreamBuf);

        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            uploadSucceeded = false;
            break;
        }

        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
```

```
                                hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

       Aws::String calculatedHash;
       if (gUseCalculatedChecksum) {
           calculatedHash = base64HashString;
       }
       Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,
                                                                        client);
       if (uploadPartOutcome.IsSuccess()) {
           const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
           Aws::S3::Model::CompletedPart completedPart;
           completedPart.SetETag(uploadPartResult.GetETag());
           completedPart.SetPartNumber(partNumber);
           switch (hashMethod) {
               case AwsDoc::S3::MD5:
                   break; // Do nothing.
               case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                   break;
               case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                   break;
               case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                   break;
               case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                   break;
               default:
                   std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                   break;
           }

           parts.push_back(completedPart);
```

```
        } else {
            std::cerr << "Error uploading part. " <<
                      uploadPartOutcome.GetError().GetMessage() << std::endl;
            uploadSucceeded = false;
            break;
        }

        uploadedBytes += bytesToRead;
        partNumber++;
    }

    if (!uploadSucceeded) {
        abortMultipartUpload(bucket, key, uploadID, client);
        return false;
    } else {

        Aws::S3::Model::CompleteMultipartUploadOutcome
 completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                key,

                                uploadID,

                                parts,

                                client);

        if (completeMultipartUploadOutcome.IsSuccess()) {
            std::cout << "Multipart upload completed." << std::endl;
            if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod))
 {
                std::cerr << "Error calculating hash." << std::endl;
                return false;
            }
        } else {
            std::cerr << "Error completing multipart upload." <<
                      completeMultipartUploadOutcome.GetError().GetMessage()
                      << std::endl;
        }

        return completeMultipartUploadOutcome.IsSuccess();
    }
}
```

```
//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            return "Default";
        case AwsDoc::S3::MD5:
            return "MD5";
        case AwsDoc::S3::SHA1:
            return "SHA1";
        case AwsDoc::S3::SHA256:
            return "SHA256";
        case AwsDoc::S3::CRC32:
            return "CRC32";
        case AwsDoc::S3::CRC32C:
            return "CRC32C";
        default:
            return "Unknown";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
*/
Aws::S3::Model::ChecksumAlgorithm
AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod)
 {
    Aws::S3::Model::ChecksumAlgorithm result =
 Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not
 valid." << std::endl;
            break;  // Default is not supported.
        case AwsDoc::S3::MD5:
            break; // Ignore MD5.
        case AwsDoc::S3::SHA1:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
            break;
```

```
        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;

    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                         const Aws::S3::S3ClientConfiguration
 &clientConfiguration) {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                               clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
```

```
  \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
            << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
 integrity workflow."
            << std::endl;
    printAsterisksLine();
    std::cout
            << "This workflow demonstrates how Amazon S3 uses checksum values to
 verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
            << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
            << "By default it calculates a checksum that is uploaded with an
 object.\n"
            << "The default checksum algorithm for PutObject and MultiPart upload
 is an MD5 hash.\n"
            << "The default checksum algorithm for TransferManager uploads is a
 CRC32 checksum."
            << std::endl;
    std::cout
            << "You can override the default behavior, requiring one of the
 following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting
 the SDK calculate the value."
              << std::endl;
    std::cout
            << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
            << std::endl;

    std::cout
            << "This workflow will locally compute checksums for files uploaded
 to an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
            << "This is done to provide demonstration code for how the checksums
 are calculated."
            << std::endl;
```

```
    std::cout << "A bucket named '" << bucketName << "' will be created for the
 object uploads."
              << std::endl;
}

//! Console interaction which explains the PutObject results.
/*!
*/
void AwsDoc::S3::explainPutObjectResults() {

    std::cout << "The upload was successful.\n";
    std::cout << "If the checksums had not matched, the upload would have
 failed."
              << std::endl;
    std::cout
            << "The checksums calculated by the server have been retrieved using
 the GetObjectAttributes."
            << std::endl;
    std::cout
            << "The locally calculated checksums have been verified against the
 retrieved checksums."
            << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
  \param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
        const Aws::String &objectKey) {
    std::cout
            << "Now the workflow will demonstrate object integrity for
 TransferManager multi-part uploads."
            << std::endl;
    std::cout
            << "The AWS C++ SDK has a TransferManager class which simplifies
 multipart uploads."
            << std::endl;
    std::cout
            << "The following code lets the TransferManager handle much of the
 checksum configuration."
            << std::endl;

    std::cout << "An object with the key '" << objectKey
```

```
              << " will be uploaded by the TransferManager using a "
              << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK
 calculate the hash value."
                  << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();
}

//! Console interaction explaining multi-part uploads.
/*!
  \param objectKey: The key for the object being uploaded.
  \param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                             HASH_METHOD chosenHashMethod) {
    std::cout
            << "Now we will provide an in-depth demonstration of multi-part
 uploading by calling the multi-part upload APIs directly."
            << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when
 uploading large files."
              << std::endl;
    std::cout
            << "In the following code, the checksums are also calculated locally
 and then compared."
            << std::endl;
    std::cout
            << "For multi-part uploads, a checksum is uploaded with each part.
 The final checksum is a concatenation of"
            << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
            << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,\""
            << " in the section \"Using part-level checksums for multipart
 uploads\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
              stringForHashMethod(chosenHashMethod) << " uploading to with object
 key\n"
```

```
              << "'" << objectKey << "'," << std::endl;

}

//! Create a large file for doing multi-part uploads.
/*!
*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
    // Generate a large file by writing this source file multiple times to a new
 file.
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                                | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
 MULTI_PART_TEST_FILE <<
                  std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in

                                   | std::ios::binary);
    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
                  std::endl;
        return false;
    }
    std::stringstream buffer;
    buffer << input.rdbuf();

    input.close();

    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();
```

```
    return true;
}
```

- 有关 API 详细信息，请参阅 AWS SDK for C++ API 参考中的以下主题。
  - AbortMultipartUpload
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - DeleteObject
  - GetObjectAttributes
  - PutObject
  - UploadPart

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 使用 Amazon S3 版本控制对象

以下代码示例展示了如何：

- 创建版本控制的 S3 存储桶。
- 获取对象的所有版本。
- 将对象回滚到以前的版本。
- 删除并还原版本控制的对象。
- 永久删除对象的所有版本。

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

创建包装 S3 操作的函数。

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
```

```
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
 expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket



def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
```

```
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
 f"{object_key}."
        )



def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest
 version
    and the object then presents as *not* deleted.
```

```
    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.",
 object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)



def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.
```

```
    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

将诗节上传到版本控制的对象并对其执行一系列操作。

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
```

```
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
    print(
        "The latest version of the stanza is now:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions =
 bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[
            f"    {version.version_id}, last modified {version.last_modified}"
            for version in obj_stanza_1_versions
        ],
        sep="\n",
    )

    # Rollback two versions.
    print("\nRolling back two versions...")
    rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
    print(
        "The latest version of the stanza:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Delete the stanza
    print("\nDeleting the stanza...")
    obj_stanza_1.delete()
    try:
        obj_stanza_1.get()
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            print("The stanza is now deleted (as expected).")
        else:
            raise
```

```
    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的以下主题。
  - CreateBucket
  - DeleteObject
  - ListObjectVersions
  - PutBucketLifecycleConfiguration

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS 开发工具包的 Amazon S3 无服务器示例

以下代码示例演示了如何将 Amazon S3 与 AWS SDK 一起使用。

示例

- 通过 Amazon S3 触发器调用 Lambda 函数

通过 Amazon S3 触发器调用 Lambda 函数

以下代码示例演示了如何实现一个 Lambda 函数，该函数接收通过将对象上传到 S3 存储桶而触发的事件。该函数从事件参数中检索 S3 存储桶名称和对象密钥，并调用 Amazon S3 API 来检索和记录对象的内容类型。

.NET

AWS SDK for .NET

> **Note**
>
> 查看 GitHub，了解更多信息。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 .NET 将 S3 事件与 Lambda 结合使用。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
```

```
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request -
 {e.Message}");

                return string.Empty;
            }
        }
    }
}
```

Go

适用于 Go V2 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Go 将 S3 事件与 Lambda 结合使用。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "log"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Printf("failed to load default config: %s", err)
  return err
 }
 s3Client := s3.NewFromConfig(sdkConfig)

 for _, record := range s3Event.Records {
  bucket := record.S3.Bucket.Name
  key := record.S3.Object.URLDecodedKey
  headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
   Bucket: &bucket,
   Key:    &key,
  })
  if err != nil {
   log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
```

```
   return err
  }
  log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
 *headOutput.ContentType)
 }

 return nil
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 将 S3 事件与 Lambda 结合使用。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
 com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
          S3EventNotificationRecord record = s3event.getRecords().get(0);
          String srcBucket = record.getS3().getBucket().getName();
          String srcKey = record.getS3().getObject().getUrlDecodedKey();

          S3Client s3Client = S3Client.builder().build();
          HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
 srcKey);

          logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
 type " + headObject.contentType());

          return "Ok";
        } catch (Exception e) {
          throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
 String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucket)
                .key(key)
                .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

JavaScript

适用于 JavaScript 的 SDK (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 JavaScript 将 S3 事件与 Lambda 结合使用。

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g,
 ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make
 sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

使用 TypeScript 将 S3 事件与 Lambda 结合使用。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
```

```
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, '
 '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure
 they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

通过 PHP 将 S3 事件与 Lambda 结合使用。

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';
```

```
class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
 $bucket . '. Make sure they exist and your bucket is in the same region as this
 function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK for Python (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Python 将 S3 事件与 Lambda 结合使用。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
 encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
 your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

通过 Ruby 将 S3 事件与 Lambda 结合使用。

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
 Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
 and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在无服务器示例存储库中查找完整示例，并了解如何进行设置和运行。

使用 Rust 将 S3 事件与 Lambda 结合使用。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};


/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
```

```
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
 SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
 name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
 exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
 contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}
```

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

故障排除

本节介绍了如何排查 Amazon S3 功能的问题，并说明了在联系 AWS Support 时如何获取您需要的请求 ID。

主题

排查 Amazon S3 中的拒绝访问（403 Forbidden）错误

当 AWS 显式或隐式拒绝授权请求时，将显示拒绝访问（HTTP 403 Forbidden）错误。

- 当策略包含特定的 AWS 操作的 Deny 语句时，将发生显式拒绝。
- 当没有适用的 Deny 语句且没有适用的 Allow 语句时，会发生隐式拒绝。

由于 AWS Identity and Access Management（IAM）策略默认情况下隐式拒绝 IAM 主体，因此该策略必须显式支持主体执行操作。否则，该策略会隐式拒绝访问。有关更多信息，请参阅《IAM 用户指南》中的显式拒绝和隐式拒绝之间的区别。有关确定是支持还是拒绝访问请求的策略评估逻辑的信息，请参阅《IAM 用户指南》中的策略评估逻辑。

以下主题涵盖了 Amazon S3 中拒绝访问错误的最常见原因。

> **Note**
>
> 对于拒绝访问（HTTP 403 Forbidden）错误，如果请求是在存储桶拥有者的个人 AWS 账户或存储桶拥有者的 AWS 组织外部发起的，则 S3 不会向存储桶拥有者收费。

主题

- 拒绝访问消息示例以及如何对其进行故障排除
- 桶策略和 IAM policy
- Amazon S3 ACL 设置
- S3 屏蔽公共访问权限设置
- Amazon S3 加密设置
- S3 对象锁定设置
- VPC 端点策略
- AWS Organizations 策略
- 接入点设置

> **Note**
>
> 如果您正在尝试对权限问题进行故障排除，请从the section called “拒绝访问消息示例以及如何对其进行故障排除”一节开始，然后转到???一节。另外，请务必遵循the section called “有关检查权限的提示”中的指导。

拒绝访问消息示例以及如何对其进行故障排除

> **Important**
>
> 从 2024 年 8 月 21 日起，针对同账户请求的增强拒绝访问错误消息将在未来几周内推出。这些消息将在所有 AWS 区域提供，包括 AWS GovCloud (US) Regions和中国区域。

大多数拒绝访问的错误消息都以 `User` *`user-arn`* `is not authorized to perform` *`action`* `on "`*`resource-arn`*`" because` *`context`* 格式显示。在此示例中，*`user-arn`* 是未获得访问权限的用户的 Amazon 资源名称（ARN），*`action`* 是策略拒绝的服务操作，而 *`resource-arn`* 是策略对其执行操作的资源的 ARN。*`context`* 字段表示有关策略类型的其它上下文，用于解释策略拒绝访问的原因。

当策略因其包含 `Deny` 语句而显式拒绝访问时，拒绝访问错误消息中将包含短语 `with an explicit deny in a` *`type`* `policy`。当策略隐式拒绝访问时，拒绝访问错误消息中将包含短语 `because no` *`type`* `policy allows the` *`action`* `action`。

Important

- 仅针对同账号请求才返回增强拒绝访问消息。跨账户请求会返回一条一般 `Access Denied` 消息。

  有关确定是支持还是拒绝跨账户访问请求的策略评估逻辑的信息，请参阅《IAM 用户指南》中的跨账户策略评估逻辑。有关说明如何授予跨账户访问权限的演练，请参阅the section called “授予跨账户权限”。
- 向目录存储桶发出的请求不会返回增强拒绝访问错误消息。目录存储桶请求会返回一条一般 `Access Denied` 消息。
- 如果同一策略类型的多个策略拒绝授权请求，拒绝访问错误消息不指定策略的数量。
- 如果多种策略类型拒绝授权请求，则错误消息仅包含其中一种策略类型。
- 如果由于多种原因而拒绝访问请求，则错误消息仅包含拒绝原因之一。

下面的示例展示了不同类型的拒绝访问错误消息的格式以及如何对每种类型的消息进行故障排除。

由于服务控制策略而拒绝访问 – 隐式拒绝

1. 检查服务控制策略（SCP）中的操作是否有缺少的 `Allow` 语句。对于以下示例，操作为 `s3:GetObject`。
2. 通过添加 `Allow` 语句来更新 SCP。有关更多信息，请参阅 AWS Organizations 用户指南中的更新 SCP。

```
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject because no service control policy allows the s3:GetObject action
```

由于服务控制策略而拒绝访问 – 显式拒绝

1. 检查服务控制策略（SCP）中的操作是否有 `Deny` 语句。对于以下示例，操作为 `s3:GetObject`。
2. 通过删除 `Deny` 语句来更新 SCP。有关更多信息，请参阅 AWS Organizations 用户指南中的更新 SCP。

```
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject with an explicit deny in a service control policy
```

由于 VPC 端点策略而拒绝访问 – 隐式拒绝

1. 检查虚拟私有云（VPC）端点策略中的操作是否有缺失的 `Allow` 语句。对于以下示例，操作为 `s3:GetObject`。
2. 通过添加 `Allow` 语句来更新 VPC 端点策略。有关更多信息，请参阅《AWS PrivateLink 指南》中的更新 VPC 端点策略。

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no VPC endpoint policy allows the s3:GetObject action
```

由于 VPC 端点策略而拒绝访问 – 显式拒绝

1. 检查虚拟私有云（VPC）端点策略中的操作是否有显式 `Deny` 语句。对于以下示例，操作为 `s3:GetObject`。
2. 通过删除 `Deny` 语句来更新 VPC 端点策略。有关更多信息，请参阅《AWS PrivateLink 指南》中的更新 VPC 端点策略。

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in a VPC endpoint policy
```

由于权限边界而拒绝访问 – 隐式拒绝

1. 检查权限边界中的操作是否有缺失的 `Allow` 语句。对于以下示例，操作为 `s3:GetObject`。
2. 通过将 `Allow` 语句添加到 IAM policy 来更新权限边界。有关更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界和编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name"
because no permissions boundary allows the s3:GetObject action
```

由于权限边界而拒绝访问 – 显式拒绝

1. 检查权限边界中的操作是否有显式 `Deny` 语句。对于以下示例，操作为 `s3:GetObject`。

2. 通过从 IAM policy 删除 Deny 语句来更新权限边界。有关更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界和编辑 IAM 策略。

```
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject with an explicit deny in a permissions boundary
```

由于会话策略而拒绝访问 – 隐式拒绝

1. 检查会话策略中的操作是否有缺失的 Allow 语句。对于以下示例，操作为 s3:GetObject。
2. 通过添加 Allow 语句来更新会话策略。有关更多信息，请参阅《IAM 用户指南》中的会话策略和编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no session policy allows the s3:GetObject action
```

由于会话策略而拒绝访问 – 显式拒绝

1. 检查会话策略中的操作是否有显式 Deny 语句。对于以下示例，操作为 s3:GetObject。
2. 通过删除 Deny 语句来更新会话策略。有关更多信息，请参阅《IAM 用户指南》中的会话策略和编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in a session policy
```

由于基于资源的策略而拒绝访问 – 隐式拒绝

> **Note**
>
> 基于资源的策略 是指诸如存储桶策略和接入点策略之类的策略。

1. 检查基于资源的策略中的操作是否有缺失的 Allow 语句。还要检查 IgnorePublicAcls S3 屏蔽公共访问权限设置是否应用于存储桶、接入点或账户级。对于以下示例，操作为 s3:GetObject。

2. 通过添加 Allow 语句来更新策略。有关更多信息，请参阅《IAM 用户指南》中的基于资源的策略和编辑 IAM 策略。

   您可能还需要调整存储桶、接入点或账户的 IgnorePublicAcls 屏蔽公共访问权限设置。有关更多信息，请参阅the section called “由于屏蔽公共访问权限设置而拒绝访问” 和the section called “配置存储桶和接入点设置”。

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no resource-based policy allows the s3:GetObject action
```

由于基于资源的策略而拒绝访问 – 显式拒绝

> **Note**
>
> 基于资源的策略 是指诸如存储桶策略和接入点策略之类的策略。

1. 检查基于资源的策略中的操作是否有显式 Deny 语句。还要检查 RestrictPublicBuckets S3 屏蔽公共访问权限设置是否应用于存储桶、接入点或账户级。对于以下示例，操作为 s3:GetObject。
2. 通过删除 Deny 语句来更新策略。有关更多信息，请参阅《IAM 用户指南》中的基于资源的策略和编辑 IAM 策略。

   您可能还需要调整存储桶、接入点或账户的 RestrictPublicBuckets 屏蔽公共访问权限设置。有关更多信息，请参阅the section called “由于屏蔽公共访问权限设置而拒绝访问” 和the section called “配置存储桶和接入点设置”。

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in a resource-based policy
```

由于基于身份的策略而拒绝访问 – 隐式拒绝

1. 检查附加到身份的基于身份的策略中的操作中是否有缺失的 Allow 语句。对于以下示例，操作为附加到用户 MaryMajor 的 s3:GetObject。
2. 通过添加 Allow 语句来更新策略。有关更多信息，请参阅《IAM 用户指南》中的基于身份的策略和编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no identity-based policy allows the s3:GetObject action
```

由于基于身份的策略而拒绝访问 – 显式拒绝

1. 检查附加到身份的基于身份的策略中的操作中是否有显式 Deny 语句。对于以下示例，操作为附加到用户 MaryMajor 的 s3:GetObject。
2. 通过删除 Deny 语句来更新策略。有关更多信息，请参阅《IAM 用户指南》中的基于身份的策略和编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in an identity-based policy
```

由于屏蔽公共访问权限设置而拒绝访问

Amazon S3 屏蔽公共访问权限特征提供接入点、存储桶和账户设置，帮助您管理对 Amazon S3 资源的公有访问。有关 Amazon S3 如何定义“公共”的更多信息，请参阅“公有”的含义。

默认情况下，新存储桶、接入点和对象不允许公有访问。但是，用户可以修改存储桶策略、接入点策略、IAM 用户策略、对象权限或访问控制列表（ACL）来支持公共访问权限。S3 屏蔽公共访问权限设置会覆盖这些策略、权限和 ACL。自 2023 年 4 月起，默认情况下为新存储桶启用所有屏蔽公共访问权限设置。

当 Amazon S3 收到访问存储桶或对象的请求时，它将确定该存储桶或存储桶拥有者的账户是否应用了屏蔽公共访问权限设置。如果请求是通过接入点发出，则 Amazon S3 还会检查接入点的屏蔽公共访问权限设置。如果现有的屏蔽公共访问权限设置禁止请求的访问，则 Amazon S3 将拒绝该请求。

Amazon S3 屏蔽公共访问权限提供四种设置。这些设置彼此独立，可任意组合使用。每个设置都可以应用于访问点、存储桶或整个 AWS 账户。如果接入点、存储桶或账户的屏蔽公共访问权限设置不同，则 Amazon S3 应用接入点、存储桶和账户设置的最严格组合。

当 Amazon S3 评估屏蔽公共访问权限设置是否禁止某一操作时，它将拒绝违反接入点、存储桶或账户设置的任何请求。

Amazon S3 屏蔽公共访问权限提供的四种设置如下：

- BlockPublicAcls – 此设置适用于 PutBucketAcl、PutObjectAcl、PutObject、CreateBucket、CopyObject 和 POST Object 请求。BlockPublicAcls 设置会导致以下行为：
  - 如果指定的访问控制列表（ACL）为公有，PutBucketAcl 和 PutObjectAcl 调用将失败。
  - 如果请求包含公有 ACL，则 PutObject 调用失败。
  - 如果将此设置应用于某个账户，则当请求包含公有 ACL 时，CreateBucket 调用将失败，并返回 HTTP 400 (Bad Request) 响应。

  例如，当对 CopyObject 请求的访问因 BlockPublicAcls 设置而被拒绝时，您将收到以下消息：

  ```
  An error occurred (AccessDenied) when calling the CopyObject operation:
  User: arn:aws:sts::123456789012:user/MaryMajor is not authorized to
  perform: s3:CopyObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name"
  because public access control lists (ACLs) are blocked by the BlockPublicAcls block
  public access setting.
  ```

- IgnorePublicAcls – IgnorePublicAcls 设置会使 Amazon S3 忽略存储桶及其包含的任何对象上的所有公有 ACL。如果您的请求的权限仅由公共 ACL 授予，则 IgnorePublicAcls 设置会拒绝该请求。

  由 IgnorePublicAcls 设置导致的任何拒绝都是隐式的。例如，如果 IgnorePublicAcls 因公有 ACL 而拒绝 GetObject 请求，您将收到以下消息：

  ```
  User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
  s3:GetObject because no resource-based policy allows the s3:GetObject action
  ```

- BlockPublicPolicy – 此设置适用于 PutBucketPolicy 和 PutAccessPointPolicy 请求。

  为存储桶设置 BlockPublicPolicy 将导致 Amazon S3 在指定的存储桶策略支持公共访问权限时拒绝对 PutBucketPolicy 的调用。如果指定的策略支持公共访问权限，则此设置也会导致 Amazon S3 针对存储桶的所有相同账户接入点拒绝对 PutAccessPointPolicy 的调用。

  如果（为接入点或底层存储桶）指定的策略支持公共访问权限，则为接入点设置 BlockPublicPolicy 会导致 Amazon S3 拒绝通过该接入点对 PutAccessPointPolicy 和 PutBucketPolicy 进行的调用。

  例如，当对 PutBucketPolicy 请求的访问因 BlockPublicPolicy 设置而被拒绝时，您将收到以下消息：

```
An error occurred (AccessDenied) when calling the PutBucketPolicy operation:
User: arn:aws:sts::123456789012:user/MaryMajor is not authorized to
perform: s3:PutBucketPolicy on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-
name"
because public policies are blocked by the BlockPublicPolicy block public
access setting.
```

- `RestrictPublicBuckets` – `RestrictPublicBuckets` 设置会将使用公有策略对接入点或存储桶的访问权限限制为仅该存储桶拥有者账户和接入点拥有者账户中的 AWS 服务主体和授权用户。此设置会阻止对接入点或存储桶的所有跨账户访问（AWS 服务主体的访问除外），但仍支持该账户内的用户管理接入点或存储桶。此设置还拒绝所有匿名（或未签名的）调用。

  由 `RestrictPublicBuckets` 设置导致的任何拒绝都是显式的。例如，如果 `RestrictPublicBuckets` 因公有存储桶或接入点策略而拒绝 `GetObject` 请求，您将收到以下消息：

  ```
  User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
  s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
  an explicit deny in a resource-based policy
  ```

有关这些设置的更多信息，请参阅 the section called “屏蔽公共访问权限设置”。要查看和更新这些设置，请参阅the section called “配置屏蔽公共访问权限”。

桶策略和 IAM policy

存储桶级别操作

如果没有存储桶策略，则存储桶会隐式支持来自存储桶拥有者账户中任何 AWS Identity and Access Management（IAM）身份的请求。桶还隐式拒绝来自任何其他账户的任何其他 IAM 身份的请求以及匿名（未签名）请求。但是，如果没有 IAM 用户策略，则会隐式拒绝请求者（除非他们是 AWS 账户根用户）发出任何请求。有关此评估逻辑的更多信息，请参阅《IAM 用户指南》中的确定是允许还是拒绝账户内的请求。

对象级别操作

如果对象归桶拥有者账户拥有，则桶策略和 IAM 用户策略在对象级别操作中的运作方式与在桶级别操作中的运行方式相同。例如，如果没有存储桶策略，则存储桶会隐式支持来自存储桶拥有者账户中任何 IAM 身份的对象请求。桶还隐式拒绝来自任何其他账户的任何其他 IAM 身份的对象请求以及匿名（未

签名）请求。但是，如果没有 IAM 用户策略，则会隐式拒绝请求者（除非他们是 AWS 账户根用户）发出任何对象请求。

如果对象归外部账户拥有，则只能通过对象访问控制列表（ACL）授予对该对象的访问权限。桶策略和 IAM 用户策略仍可用于拒绝对象请求。

因此，为确保存储桶策略或 IAM 用户策略不会导致拒绝访问（403 禁止）错误，请确保满足以下要求：

- 对于同账户访问，无论是在桶策略还是 IAM 用户策略中，都不得针对您尝试向其授予权限的请求者使用显式 Deny 语句。如果您想仅使用桶策略和 IAM 用户策略授予权限，则其中一个策略中必须至少有一条显式 Allow 语句。
- 对于跨账户访问，无论是在存储桶策略还是 IAM 用户策略中，都不得针对您尝试向其授予权限的请求者使用显式 Deny 语句。要仅使用存储桶策略和 IAM 用户策略授予跨账户权限，请确保请求者的存储桶策略和 IAM 用户策略都包含显式 Allow 语句。

> **Note**
>
> 桶策略中的 Allow 语句仅适用于由同一桶拥有者账户拥有的对象。但是，无论对象所有权如何，桶策略中的 Deny 语句都适用于所有对象。

查看或编辑您的桶策略

> **Note**
>
> 要查看或编辑桶策略，您必须拥有 s3:GetBucketPolicy 权限。

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 从桶列表中，选择要查看或编辑其桶策略的桶的名称。
4. 选择 Permissions（权限）选项卡。
5. 在 Bucket policy（存储桶策略）下，请选择 Edit（编辑）。将出现 Edit bucket policy（编辑存储桶策略）页面。

要使用 AWS Command Line Interface（AWS CLI）查看或编辑您的桶策略，请使用 get-bucket-policy 命令。

> **Note**
>
> 如果您因为存储桶策略不正确而被锁定在存储桶之外，sign in to the AWS Management Console by using your AWS 账户 root user credentials（请使用 Amazon 账户根用户凭证登录 Amazon 管理控制台）。要重新获得对存储桶的访问权限，请务必使用 AWS 账户根用户凭证删除不正确的存储桶策略。

有关检查权限的提示

要检查请求者是否具有执行 Amazon S3 操作的适当权限，请尝试以下操作：

- 确定请求者。如果是未签名的请求，则它是没有 IAM 用户策略的匿名请求。如果这是使用预签名 URL 的请求，则用户策略会与对请求进行签名的 IAM 用户或角色的策略相同。
- 验证您使用的是正确的 IAM 用户或角色。您可以通过检查 AWS Management Console的右上角或使用 aws sts get-caller-identity 命令来验证您的 IAM 用户或角色。
- 检查与 IAM 用户或角色相关的 IAM policy。您可以使用以下方法之一：
  - 使用 IAM policy simulator 测试 IAM policy。
  - 查看不同的 IAM policy 类型。
- 如果需要，编辑您的 IAM 用户策略。
- 查看以下显式拒绝或允许访问的策略示例：
  - 显式允许 IAM 用户策略：IAM：允许和拒绝以编程方式和在控制台中访问多个服务
  - 显式允许桶策略：授予多个账户上传对象或设置对象 ACL 以进行公共访问的权限
  - 显式拒绝 IAM 用户策略：AWS：根据请求的 AWS 区域拒绝访问 AWS
  - 显式拒绝桶策略：要求对写入桶的所有对象使用 SSE-KMS

Amazon S3 ACL 设置

检查 ACL 设置时，请先查看您的对象所有权设置，以检查桶上是否已启用 ACL。请注意，ACL 权限只能用于授予权限，而不能用于拒绝请求。对于由存储桶策略或 IAM 用户策略中的显式拒绝而被拒绝的请求者，ACL 也不能用于向他们授予访问权限。

对象所有权设置设为强制桶拥有者

如果启用了强制桶拥有者设置，则 ACL 设置不太可能导致拒绝访问（403 禁止）错误，因为此设置会禁用适用于桶和对象的所有 ACL。强制桶拥有者是 Amazon S3 桶的原定设置，并且是推荐设置。

对象所有权设置设为首选桶拥有者或对象写入者

使用首选桶拥有者设置或对象编写者设置时，ACL 权限仍然有效。ACL 有两种：桶 ACL 和对象 ACL。有关这两种类型的 ACL 之间的区别，请参阅 ACL 权限和访问策略权限的映射。

根据被拒绝的请求的操作，检查您的桶或对象的 ACL 权限：

- 如果 Amazon S3 拒绝了 `LIST`、`PUT` 对象、`GetBucketAcl` 或 `PutBucketAcl` 请求，请查看您的桶的 ACL 权限。

  > **Note**
  >
  > 您无法使用存储桶 ACL 设置授予 `GET` 对象权限。

- 如果 Amazon S3 拒绝了对 S3 对象的 `GET` 请求或拒绝了 PutObjectAcl 请求，请查看该对象的 ACL 权限。

  > **Important**
  >
  > 如果拥有该对象的账户与拥有该桶的账户不同，则对该对象的访问权限不受桶策略控制。

排查在跨账户对象所有权期间来自 **GET** 对象请求的拒绝访问（403 禁止）错误

查看桶的对象所有权设置以确定对象拥有者。如果您有权访问对象 ACL，则还可以检查对象拥有者的账户。（要查看对象拥有者的账户，请在 Amazon S3 控制台中查看对象 ACL 设置。）或者，您也可以发出 `GetObjectAcl` 请求，以查找对象拥有者的规范 ID 来验证对象拥有者账户。原定设置情况下，ACL 为对于对象拥有者账户的 `GET` 请求授予显式允许权限。

确认对象拥有者与桶拥有者不同后，根据您的使用案例和访问级别，选择以下方法之一来帮助解决拒绝访问（403 禁止）错误：

- 禁用 ACL（推荐）– 此方法适用于所有对象，可由桶拥有者执行。这种方法自动向桶拥有者授予对桶中每个对象的所有权和完全控制权。在实施此方法之前，请检查禁用 ACL 的先决条件。有关如何将您的桶设置为强制桶拥有者（推荐）模式的信息，请参阅在现有桶上设置对象所有权。

> **Important**
>
> 为防止拒绝访问（403 禁止）错误，请务必在禁用 ACL 之前将 ACL 权限迁移到桶策略。有关更多信息，请参阅从 ACL 权限迁移的桶策略示例。

- 将对象拥有者更改为桶拥有者 - 此方法可以应用于单个对象，但只有对象拥有者（或具有相应权限的用户）才能更改对象的所有权。可能会收取额外的 PUT 费用。（有关更多信息，请参阅 Amazon S3 定价。）此方法向桶拥有者授予对象的完全所有权，允许桶拥有者通过桶策略控制对于对象的访问权限。

  要更改对象的所有权，请执行以下操作之一：

  - 您（桶拥有者）可以将对象复制回桶。
  - 您可以将桶的对象所有权设置更改为首选桶拥有者。如果禁用版本控制，则桶中的对象将被覆盖。如果启用了版本控制，则同一对象的重复版本将出现在桶中，而桶拥有者可以将生命周期规则设置为过期。有关如何更改对象所有权设置的说明，请参阅为现有存储桶设置对象所有权。

    > **Note**
    >
    > 当您将对象所有权设置更新为首选桶拥有者时，该设置仅适用于上传到桶的新对象。

  - 您可以让对象拥有者使用 `bucket-owner-full-control` 标准对象 ACL 再次上传对象。

  > **Note**
  >
  > 对于跨账户上传，您还可以在桶策略中要求使用 `bucket-owner-full-control` 标准对象 ACL。有关桶策略示例，请参阅在授予上传对象的跨账户权限的同时，确保桶拥有者拥有完全控制权。

- 将对象编写者保持为对象拥有者 - 此方法不会更改对象拥有者，但它允许您单独授予对于对象的访问权限。要授予对于对象的访问权限，您必须拥有该对象的 `PutObjectAcl` 权限。然后，要修复“拒绝访问（403 禁止）”错误，请将请求者添加为被授予者，以访问对象的 ACL 中的对象。有关更多信息，请参阅 配置 ACL。

S3 屏蔽公共访问权限设置

如果失败的请求涉及公共访问权限或公共策略，请检查您的账户、存储桶或接入点上的 S3 屏蔽公共访问权限设置。有关排查与 S3 屏蔽公共访问权限设置相关的拒绝访问错误的更多信息，请参阅the section called “由于屏蔽公共访问权限设置而拒绝访问”。

Amazon S3 加密设置

Amazon S3 支持对您的桶进行服务器端加密。服务器端加密是指由接收数据的应用程序或服务在目标位置对数据进行加密。Amazon S3 在将您的数据写入 AWS 数据中心内的磁盘时会在对象级别加密这些数据，并在您访问这些数据时解密这些数据。

原定设置情况下，Amazon S3 现在应用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3），作为 Amazon S3 中每个桶的基本加密级别。Amazon S3 还允许您在上传对象时指定服务器端加密方法。

查看您的桶的服务器端加密状态和加密设置

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 从桶列表中，请选择要检查其加密设置的桶。
4. 选择属性选项卡。
5. 向下滚动到原定设置加密部分，并查看加密类型设置。

要使用 AWS CLI 检查您的加密设置，请使用 get-bucket-encryption 命令。

检查对象的加密状态

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 从桶列表中，请选择包含对象的桶的名称。
4. 从对象列表中，请选择要为其添加或更改加密的对象的名称。

   此时将显示对象的详细信息页面。
5. 向下滚动到服务器端加密设置部分，以查看对象的服务器端加密设置。

要使用 AWS CLI 检查您的对象加密状态，请使用 head-object 命令。

加密和权限要求

Amazon S3 支持三种类型的服务器端加密：

- 具有 Amazon S3 托管密钥的服务器端加密（SSE-S3）
- 具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS）
- 具有客户提供密钥的服务器端加密（SSE-C）

根据您的加密设置，请确保满足以下权限要求：

- SSE-S3 - 不需要额外的权限。
- SSE-KMS（使用客户自主管理型密钥）- 要上传对象，需要针对 AWS KMS key 的 `kms:GenerateDataKey` 权限。要下载对象和执行对象的分段上传，需要针对 KMS 密钥的 `kms:Decrypt` 权限。
- SSE-KMS（具有 AWS 托管式密钥）– 请求者必须来自拥有 `aws/s3` KMS 密钥的同一个账户。请求者还必须具有正确的 Amazon S3 权限才能访问该对象。
- SSE-C（具有客户提供的密钥）- 无需其他权限。您可以配置桶策略，以要求和限制具有客户提供的加密密钥的服务器端加密来加密桶中的对象。

如果使用客户自主管理型密钥加密对象，请确保 KMS 密钥策略允许您执行 `kms:GenerateDataKey` 或 `kms:Decrypt` 操作。有关检查 KMS 密钥策略的说明，请参阅《AWS Key Management Service 开发人员指南》中的查看密钥策略。

S3 对象锁定设置

如果您的桶启用了 S3 对象锁定并且该对象受保留期或法定保留保护，则当您尝试删除该对象时，Amazon S3 会返回拒绝访问（403 禁止）错误。

检查桶是否启用了对象锁定

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 从桶列表中，请选择您想要查看的桶的名称。

4. 选择属性选项卡。
5. 向下滚动到对象锁定部分。验证对象锁定设置是已启用还是已禁用。

要确定该对象是受保留期保护还是受法定保留保护，请查看对象的锁定信息。

如果对象受保留期或法定保留保护，请检查以下各项：

- 如果对象版本受合规性保留模式保护，则无法将其永久删除。来自任何请求者（包括 AWS 账户根用户）的永久 DELETE 请求都将导致拒绝访问（403 禁止）错误。另请注意，当您对于受合规性保留模式保护的对象提交 DELETE 请求时，Amazon S3 会为该对象创建删除标记。
- 如果对象版本受监管保留模式保护并且您具有 `s3:BypassGovernanceRetention` 权限，则可以绕过此保护并永久删除该版本。有关更多信息，请参阅绕过监管模式。
- 如果对象版本受法定保留保护，则永久 DELETE 请求可能会导致拒绝访问（403 禁止）错误。要永久删除对象版本，必须解除对于对象版本的法定保留。要解除法定保留，您必须具有 `s3:PutObjectLegalHold` 权限。有关解除法定保留的更多信息，请参阅配置 S3 对象锁定。

VPC 端点策略

如果您使用虚拟私有云（VPC）端点访问 Amazon S3，请确保 VPC 端点策略不会阻止您访问 Amazon S3 资源。原定设置情况下，VPC 端点策略允许向 Amazon S3 发出的所有请求。您还可以配置 VPC 端点策略以限制某些请求。有关如何检查 VPC 端点策略的信息，请参阅《AWS PrivateLink 指南》中的使用端点策略控制对 VPC 端点的访问。

AWS Organizations 策略

如果您的 AWS 账户属于某个组织，则 AWS Organizations 策略可能会阻止您访问 Amazon S3 资源。默认情况下，AWS Organizations 策略不会阻止向 Amazon S3 发出的任何请求。但是，请确保您的 AWS Organizations 策略未配置为阻止对 S3 桶进行访问。有关如何检查 AWS Organizations 策略的说明，请参阅《AWS Organizations 用户指南》中的列出所有策略。

接入点设置

如果您在通过 Amazon S3 接入点发出请求时收到“拒绝访问（403 禁止）”错误，则可能需要检查以下内容：

- 接入点的配置
- 用于接入点的 IAM 用户策略

- 用于管理或配置跨账户接入点的桶策略

接入点配置和策略

- 创建接入点时，可以选择将互联网或 VPC 指定为网络来源。如果网络来源设置为仅限 VPC，Amazon S3 将拒绝向接入点发出的任何并非源自指定 VPC 的请求。要检查接入点的网络来源，请参阅创建限制到 Virtual Private Cloud 的接入点。
- 使用接入点，您还可以配置自定义的屏蔽公共访问权限设置，其工作原理与桶或账户级别的屏蔽公共访问权限设置类似。要查看您的自定义屏蔽公共访问权限设置，请参阅管理接入点的公有访问。
- 要使用接入点向 Amazon S3 发出成功的请求，请确保请求者拥有必要的 IAM 权限。有关更多信息，请参阅 配置使用接入点的 IAM 策略。
- 如果请求涉及跨账户接入点，请确保桶拥有者已更新桶策略，以授权来自接入点的请求。有关更多信息，请参阅 授予跨账户接入点的权限。

如果在检查了本主题中的所有项目后，拒绝访问（403 禁止）错误仍然存在，请检索您的 Amazon S3 请求 ID 并联系 AWS Support 以获取更多指导。

排查批量操作问题

以下主题列出了常见错误，以帮助您排除在批量操作过程中可能遇到的问题。

常见错误

- 存在权限问题或启用了 S3 对象锁定保留模式时，未交付作业报告
- S3 批量复制失败并出现错误：生成清单时未找到符合筛选条件的密钥
- 向现有复制配置添加新的复制规则后，会出现批量操作失败
- 对于对象的批量操作失败，错误为 400 InvalidRequest：由于缺少 VersionId，任务失败
- 在启用任务标签选项的情况下创建任务失败
- 读取清单的访问被拒绝

存在权限问题或启用了 S3 对象锁定保留模式时，未交付作业报告

在缺少必需权限或者对目标存储桶启用了对象锁定保留模式（监管模式或合规性模式）时，会发生以下错误。

错误：失败的原因。无法将任务报告写入您的报告桶。请检查您的权限。

IAM 角色和信任策略必须配置为允许 S3 批量操作访问，以在要交付报告的存储桶中 PUT 对象。如果缺少这些必需的权限，则会出现作业报告交付失败的情况。

启用保留模式后，桶受到“一次写入，多次读取”（WORM）保护。在目标存储桶上，不支持在启用了保留模式的时进行对象锁定，因此作业完成报告交付尝试失败。要解决此问题，请为您的任务完成报告选择一个未启用对象锁定保留模式的目标桶。

S3 批量复制失败并出现错误：生成清单时未找到符合筛选条件的密钥

错误：生成清单时未找到符合筛选条件的密钥。

出现此错误的原因为以下之一：

- 当源存储桶中的对象存储在 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 存储类中时。

  要对这些对象使用批量复制，请先在批量操作任务中使用 S3 启动还原对象操作，将它们还原到 S3 Standard 存储类。有关更多信息，请参阅恢复已归档的对象和还原对象（批量操作）。还原对象后，您可以使用批量复制任务来复制它们。
- 当提供的筛选条件与源存储桶中的任何有效对象都不匹配时。

  验证并更正筛选条件。例如，在批量复制规则中，筛选条件是查找 *amzn-s3-demo-bucket* 中带有 `Tax/` 前缀的所有对象。如果前缀名称输入不准确，开头和结尾都有一个斜杠（即 `/Tax/`），而不是仅在结尾处有斜杠，则找不到任何 S3 对象。要解决此错误，请更正前缀，在本例中就是将复制规则中的 `/Tax/` 更改为 `Tax/`。

向现有复制配置添加新的复制规则后，会出现批量操作失败

批量操作尝试对源桶的复制配置中的每条规则执行现有对象复制。如果任何现有复制规则出现问题，则可能会出现失败。

批量操作任务的完成报告解释了任务失败的原因。有关常见错误的列表，请参阅Amazon S3 复制失败原因。

对于对象的批量操作失败，错误为 400 InvalidRequest：由于缺少 VersionId，任务失败

如果批量操作任务正在对受版本控制的桶中的对象执行操作，并在清单中遇到版本 ID 字段为空的对象，则会出现以下示例错误。

错误：*BUCKET_NAME,prefix/file_name*,失败,400,InvalidRequest,由于缺少 VersionId，任务失败

之所以出现此错误，是因为清单中的版本 ID 字段是空字符串，而不是文本 `null` 字符串。

这一特定对象或这些对象的批量操作将失败，但整个任务不会失败。如果在操作期间将清单格式配置为使用版本 ID，则会出现此问题。非版本控制的任务不会遇到此问题，因为它们仅对每个对象的最新版本进行操作，并忽略清单中的版本 ID。

要解决此问题，请将空版本 ID 转换为 `null` 字符串。有关更多信息，请参阅 the section called “将空版本 ID 字符串转换为空字符串”。

在启用任务标签选项的情况下创建任务失败

如果没有 `s3:PutJobTagging` 权限，在启用了任务标签选项的情况下创建批量操作任务会导致 `403 access denied` 错误。

要在启用任务标签选项的情况下创建批量操作任务，创建批量操作任务的 AWS Identity and Access Management（IAM）用户除 `s3:CreateJob` 权限外还必须具有 `s3:PutJobTagging` 权限。

有关批量操作所需权限的更多信息，请参阅the section called “授予权限”。

读取清单的访问被拒绝

如果您在尝试创建批量操作任务时，批量操作无法读取清单文件，则可能会出现以下错误。

AWS CLI

禁止读取清单失败的原因：AccessDenied

Amazon S3 控制台

警告：无法获取清单对象的 ETag。指定其他对象以继续。

要解决此问题，请执行下列操作：

- 验证您用于创建批量操作任务的 AWS 账户的 IAM 角色是否具有 `s3:GetObject` 权限。账户的 IAM 角色必须具有允许批量操作读取清单文件的 `s3:GetObject` 权限。

  有关批量操作所需权限的更多信息，请参阅the section called “授予权限”。
- 检查清单对象的元数据中是否存在与 S3 对象所有权不匹配的访问权限。有关 S3 对象所有权的更多信息，请参阅the section called “控制对象所有权”。

- 检查 AWS Key Management Service（AWS KMS）密钥是否用于加密清单文件。

  S3 批量操作支持用 AWS KMS 加密的 CSV 清单报告。但是，S3 批量操作不支持用 AWS KMS 加密的 CSV 清单文件。有关更多信息，请参阅配置 Amazon S3 清单 和指定清单。

排查 Amazon S3 生命周期问题

以下信息可以帮助您排查 Amazon S3 生命周期规则的常见问题。

主题

我对我的桶运行了一个列表操作，看到了我认为已按生命周期规则过期或转换的对象。

S3 生命周期对象转换和对象过期是异步操作。因此，在对象符合过期或转换条件的时间与它们实际转换或过期的时间之间可能会有延迟。一旦满足生命周期规则，即使操作尚未完成，也会立即应用账单的变化。此行为的例外情况是，如果您有一个生命周期规则集转换为 S3 Intelligent-Tiering 存储类。在此类情况下，在对象转换为 S3 Intelligent-Tiering 存储类之前，账单不会发生更改。有关账单变化的更多信息，请参阅设置桶的生命周期配置。

> **Note**
>
> Amazon S3 不会将小于 128KB 的对象从 S3 Standard 或 S3 Standard-IA 存储类转换为 S3 Intelligent-Tiering、S3 Standard-IA 或 S3 One Zone-IA 存储类。

如何监控生命周期规则执行的操作？

要监控生命周期规则执行的操作，可以使用以下功能：

- S3 事件通知：您可以设置 S3 事件通知，以便收到有关任何 S3 生命周期到期或转换事件的通知。
- S3 服务器访问日志：您可以为 S3 存储桶启用服务器访问日志，以捕获 S3 生命周期操作，例如对象转换到另一个存储类或对象到期。有关更多信息，请参阅生命周期和日志记录。

要每天查看由生命周期操作引起的存储更改，我们建议使用 S3 Storage Lens 存储统计管理工具控制面板，而不是使用 Amazon CloudWatch 指标。在 Storage Lens 存储分析功能控制面板中，可以查看以下指标，这些指标监控对象计数或大小：

- 当前版本字节数
- 当前版本对象计数
- 非当前版本字节数
- 非当前版本对象计数
- 删除标记对象计数
- 删除标记存储字节数
- 未完成的分段上传字节数
- 未完成的分段上传对象计数

即使在启用了版本控制的存储桶上设置了生命周期规则之后，我的 S3 对象计数仍增加。

在启用了版本控制的存储桶中，当对象到期时，不会从存储桶中完全删除该对象。而是创建删除标记作为对象的最新版本。删除标记仍算作对象。因此，如果创建的生命周期规则仅使当前版本过期，则 S3 桶中的对象计数实际上会增加而不是减少。

例如，假设一个包含 100 个对象的 S3 桶启用了版本控制，并且生命周期规则设置为在 7 天后使对象的当前版本过期。第七天之后，对象计数增加到 200，这是因为除了 100 个原始对象（现在是非当前版本）之外，还创建了 100 个删除标记。有关启用版本控制的桶的 S3 生命周期配置规则操作的更多信息，请参阅对于桶设置生命周期配置。

要永久删除对象，请添加额外的生命周期配置以删除对象的先前版本、过期的删除标记和未完成的分段上传。有关如何创建新的生命周期规则的说明，请参阅对于桶设置生命周期配置。

> **Note**
>
> - Amazon S3 将对象的转换或过期日期舍入到第二天的世界标准时间午夜。
>
> 在评估对象的生命周期操作时，Amazon S3 使用以 UTC 表示的对象创建时间。例如，假设一个不受版本控制的存储桶，其生命周期规则配置为在一天后对象就会到期。假设一个对象是在太平洋夏令时（PDT）的 1 月 1 日 17:05 创建的，该时间对应于 UTC 时间 1 月 2 日 00:05。该对象在 UTC 时间 1 月 3 日 00:05 时已存在一天，因此，当 S3 生命周期在 UTC 时间 1 月 4 日 00:00 评估对象时，该对象符合到期条件。
>
> 由于 Amazon S3 生命周期操作是异步发生的，因此，在生命周期规则中指定的日期与对象的实际物理转换之间可能会有一些延迟。有关更多信息，请参阅转换或到期延迟。
>
> 有关更多信息，请参阅生命周期规则：基于对象的存在期限。
> - 对于受对象锁定保护的 S3 对象，不会永久删除当前版本。而是向对象添加删除标记，使其成为非当前对象。非当前版本将被保留，并且不会永久过期。

如何使用生命周期规则清空 S3 桶？

S3 生命周期规则是清空包含数百万个对象的 S3 桶的有效工具。要从 S3 存储桶中删除大量对象，请确保使用以下两对生命周期规则：

- 将对象的当前版本设为过期和永久删除对象的先前版本
- 删除过期的删除标记和删除未完成的分段上传

有关如何创建生命周期配置规则的步骤，请参阅对于桶设置生命周期配置。

> **Note**
>
> 对于受对象锁定保护的 S3 对象，不会永久删除当前版本。而是向对象添加删除标记，使其成为非当前对象。非当前版本将被保留，并且不会永久过期。

在将对象转换到成本较低的存储类后，我的 Amazon S3 账单增加了。

将对象转换为成本较低的存储类后，您的账单可能会增加，原因有很多：

- 小型对象的 S3 Glacier 开销费用

  对于每个转换为 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 的对象，与此存储更新关联的总开销为 40KB。作为 40KB 开销的一部分，8KB 用于存储元数据和对象的名称。此 8KB 按照 S3 Standard 费率收费。剩余的 32KB 用于编制索引和相关的元数据。这 32KB 按照 S3 Glacier Flexible Retrieval 或 S3 Glacier Deep Archive 定价收费。

  因此，如果您要存储许多大小较小的对象，我们不建议使用生命周期转换。相反，为了减少任何开销费用，请考虑在将许多较小的对象存储到 Amazon S3 中之前，将它们聚合为数量较少的大型对象。有关成本注意事项的更多信息，请参阅转换为 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类（对象存档）。

- 最低存储费用

  某些 S3 存储类有最短存储持续时间要求。在满足最短持续时间之前从这些类中删除、覆盖或转换的对象将按比例收取提前转换或删除费用。这些最短存储持续时间要求如下所示：

  - S3 Standard-IA 和 S3 One Zone-IA – 30 天
  - S3 Glacier Flexible Retrieval 和 S3 Glacier Instant Retrieval – 90 天
  - S3 Glacier Deep Archive – 180 天

  有关这些要求的更多信息，请参阅使用 S3 生命周期转换对象的约束部分。有关一般 S3 定价信息，请参阅 Amazon S3 定价和 AWS 定价计算器。

- 生命周期转换成本

  每次通过生命周期规则将对象转换到不同的存储类时，Amazon S3 都会将该转换视为一个转换请求。这些转换请求的成本未包含在这些存储类的成本中。如果计划转换大量对象，请在转换到较低层时考虑请求成本。有关更多信息，请参阅 Amazon S3 定价。

我已经更新了我的桶策略，但我的 S3 对象仍会被过期的生命周期规则删除。

桶策略中的 `Deny` 语句不会阻止在生命周期规则中定义的对象过期。生命周期操作（例如转换或过期）不使用 S3 `DeleteObject` 操作。相反，S3 生命周期操作是使用内部 S3 端点执行的。（有关更多信息，请参阅生命周期和日志记录。）

要防止您的生命周期规则采取任何操作，您必须编辑、删除或禁用该规则。

我能否恢复由 S3 生命周期规则过期的 S3 对象？

恢复由 S3 生命周期过期的对象的唯一方法是通过版本控制，版本控制必须在对象符合过期条件之前就位。您无法撤消由生命周期规则执行的过期操作。如果现有的 S3 生命周期规则永久删除了对象，则无法恢复这些对象。要对桶启用版本控制，请参阅the section called “使用 S3 版本控制”。

如果您已对桶应用了版本控制，并且对象的非当前版本仍然完好无损，则可以恢复过期对象的先前版本。有关 S3 生命周期规则操作的行为和版本控制状态的更多信息，请参阅描述生命周期操作的元素中的生命周期操作和桶版本控制状态表。

> **Note**
>
> 如果 S3 桶受 AWS Backup 或 S3 复制保护，则您也可以使用这些功能来恢复过期的对象。

如何从我的生命周期规则中排除前缀？

S3 生命周期不支持在规则中排除前缀。而是，使用标签来标记要包含在规则中的所有对象。有关在生命周期规则中使用标签的更多信息，请参阅the section called “示例 1：指定筛选条件”。

如何在我的生命周期规则中包含多个前缀？

S3 生命周期不支持在规则中包含多个前缀。而是，使用标签来标记要包含在规则中的所有对象。有关在生命周期规则中使用标签的更多信息，请参阅the section called “示例 1：指定筛选条件”。

但是，如果您有一个或多个前缀以相同的字符开头，则可以通过在筛选条件中指定不带尾部斜杠 (/) 的部分前缀，来将所有这些前缀包含在规则中。例如，假设您有以下前缀：

```
sales1999/
sales2000/
sales2001/
```

要在规则中包含所有三个前缀，请在生命周期规则中指定 `<Prefix>sales</Prefix>`。

对复制进行问题排查

本节列出了 Amazon S3 复制的问题排查提示以及有关 S3 批量复制错误的信息。

主题

- S3 复制问题排查提示

- 批量复制错误

S3 复制问题排查提示

如果在您配置复制之后，对象副本未出现在目标桶中，请使用以下问题排查提示确定并修复问题。

- 大多数对象会在 15 分钟内复制。Amazon S3 复制对象所需的时间取决于多个因素，包括源和目标区域对，以及对象的大小。对于大型对象，复制可能需要几个小时。要想了解复制时间，您可以使用 S3 Replication Time Control（S3 RTC）。

  如果要复制的对象较大，请稍候片刻，然后再检查它是否出现在目标桶中。还可以检查源对象的复制状态。如果对象复制状态为 PENDING，表示 Amazon S3 尚未完成复制。如果对象复制状态为 FAILED，请检查对源桶设置的复制配置。此外，要在复制期间接收有关失败的信息，您可以设置 Amazon S3 事件通知来接收复制失败事件。有关更多信息，请参阅使用 Amazon S3 事件通知接收复制失败事件。
- 您可以调用 HeadObject API 操作来检查对象的复制状态。HeadObject API 操作返回对象的 PENDING、COMPLETED 或 FAILED 复制状态。在对 HeadObject API 调用的响应中，复制状态将在 x-amz-replication-status 元素中返回。

  > **Note**
  >
  > 要运行 HeadObject，您必须对您请求的对象拥有读取权限。HEAD 请求与 GET 请求具有相同的选项，无需执行 GET 操作。例如，要使用 AWS Command Line Interface（AWS CLI）运行 HeadObject 请求，可以运行以下命令。将 *user input placeholders* 替换为您自己的信息。
  >
  > ```
  > aws s3api head-object --bucket my-bucket --key index.html
  > ```
- HeadObject 返回具有 FAILED 复制状态的对象后，您可以使用 S3 批量复制来复制这些失败的对象。或者，您可以将失败的对象重新上传到源桶，这将启动新对象的复制。
- 在源桶上的复制配置中，验证以下几点：
  - 目标存储桶的 Amazon 资源名称 (ARN) 是否正确。
  - 键名前缀是否正确。例如，如果将配置设置为复制具有前缀 Tax 的对象，则仅复制具有 Tax/document1 或 Tax/document2 等键名的对象。不会复制具有键名 document3 的对象。
  - 复制规则的状态为 Enabled。
- 在复制配置中验证没有在任何桶上暂停版本控制。源桶和目标桶必须均已启用版本控制。

- 如果将复制规则设置为将对象所有权更改为目标桶拥有者，则用于复制的 AWS Identity and Access Management（IAM）角色必须具有 `s3:ObjectOwnerOverrideToBucketOwner` 权限。此权限是对资源（在本例中为目标桶）授予的。例如，以下 `Resource` 语句显示如何授予对目标桶的这一权限：

```
{
  "Effect":"Allow",
  "Action":[
    "s3:ObjectOwnerOverrideToBucketOwner"
  ],
  "Resource":"arn:aws:s3:::DestinationBucket/*"
}
```

- 如果目标桶由另一个账户拥有，则目标桶的拥有者还必须通过目标桶策略向源桶拥有者授予 `s3:ObjectOwnerOverrideToBucketOwner` 权限。要使用以下示例桶策略，请将 *`user input placeholders`* 替换为您自己的信息：

```
{
  "Version": "2012-10-17",
  "Id": "Policy1644945280205",
  "Statement": [
    {
      "Sid": "Stmt1644945277847",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateTags",
        "s3:ObjectOwnerOverrideToBucketOwner"
      ],
      "Resource": "arn:aws:s3:::DestinationBucket/*"
    }
  ]
}
```

> **Note**
>
> 如果目标桶的对象所有权设置包括强制桶拥有者，则无需在复制规则中将此设置更新为将对象所有权更改为目标桶拥有者。原定设置情况下，对象所有权将发生变化。有关更改副本所有权的更多信息，请参阅更改副本所有权。

- 如果您要在跨账户方案（即源桶和目标桶由不同的 AWS 账户拥有）中设置复制配置，则目标桶无法配置为申请方付款桶。有关更多信息，请参阅使用申请方付款存储桶进行存储传输和使用。
- 如果桶的源对象使用 AWS Key Management Service（AWS KMS）密钥进行加密，则必须将复制规则配置为包含 AWS KMS 加密的对象。确保在 Amazon S3 控制台中的加密设置下选择复制使用 AWS KMS 加密的对象。然后，选择用于加密目标对象的 AWS KMS 密钥。

> **Note**
>
> 如果目标桶位于其他账户中，请指定目标账户拥有的 AWS KMS 客户自主管理型密钥。不要使用原定设置的 Amazon S3 托管式密钥（aws/s3）。使用原定设置密钥会使用由源账户拥有的 Amazon S3 托管式密钥加密对象，从而防止与其他账户共享对象。因此，目标账户将无法访问目标桶中的对象。

要使用属于目标账户的 AWS KMS 密钥加密目标对象，目标账户必须在 KMS 密钥策略中向复制角色授予 kms:GenerateDataKey 和 kms:Encrypt 权限。要在 KMS 密钥策略中使用以下示例语句，请将 *user input placeholders* 替换为您自己的信息：

```
{
    "Sid": "AllowS3ReplicationSourceRoleToUseTheKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
    },
    "Action": ["kms:GenerateDataKey", "kms:Encrypt"],
    "Resource": "*"
}
```

如果您在 AWS KMS 密钥策略中的 Resource 语句中使用星号（*），则该策略将使用 KMS 密钥的权限仅授予复制角色。该策略不允许复制角色提升其权限。

原定设置情况下，KMS 密钥策略向根用户授予对密钥的完全权限。这些权限可以委派给同一账户中的其他用户。除非源 KMS 密钥策略中具有 Deny 语句，否则，使用 IAM policy 向复制角色授予对源 KMS 密钥的权限就足够了。

> **Note**
>
> 将访问限制到特定 CIDR 范围、VPC 端点或 S3 接入点的 KMS 密钥策略可能会导致复制失败。

如果源或目标 KMS 密钥根据加密上下文授予权限，请确认对于桶开启了 Amazon S3 桶密钥。如果桶已开启了 S3 桶密钥，则加密上下文必须是桶级资源，如下所示：

```
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::SOURCE_BUCKET_NAME"
     ]
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::DESTINATION_BUCKET_NAME"
     ]
```

除了 KMS 密钥策略授予的权限外，源账户还必须向复制角色的 IAM policy 添加以下最低权限：

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": [
        "SourceKmsKeyArn"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey",
        "kms:Encrypt"
    ],
    "Resource": [
        "DestinationKmsKeyArn"
```

```
    ]
}
```

有关如何复制使用 AWS KMS 加密的对象的更多信息，请参阅复制加密的对象。

- 如果目标桶由另一个 AWS 账户拥有，请验证桶拥有者是否对目标桶设置了允许源桶拥有者复制对象的桶策略。有关示例，请参阅当源存储桶和目标存储桶由不同账户拥有时配置复制。
- 如果在验证权限后您的对象仍无法复制，请检查以下位置是否存在任何显式 Deny 语句：
  - 源或目标桶策略中的 Deny 语句。如果桶策略拒绝针对以下任一操作访问复制角色，则复制将失败：

    源桶：

    ```
            "s3:GetReplicationConfiguration",
            "s3:ListBucket",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
    ```

    目标桶：

    ```
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
    ```

  - 附加到 IAM 角色的 Deny 语句或权限边界可能导致复制失败。
  - 附加到源账户或目标账户的 AWS Organizations 服务控制策略中的 Deny 语句可能导致复制失败。
- 如果对象副本未出现在目标桶中，以下问题可能会禁止复制：
  - 若源桶中的某对象是另一个复制配置所创建的副本，Amazon S3 不会复制该对象。例如，如果您设置从桶 A 到桶 B 再到桶 C 的复制配置，则 Amazon S3 不会将桶 B 中的对象副本复制到桶 C。
  - 源桶拥有者可以向其他 AWS 账户授予上载对象的权限。默认情况下，源桶拥有者对于其他账户创建的对象没有权限。复制配置仅复制源桶拥有者对其具有访问权限的对象。源桶拥有者可以向其他 AWS 账户授予有条件地创建对象的权限，从而要求针对这些对象的显式访问权限。有关策略示例，请参阅在授予上传对象的跨账户权限的同时，确保存储桶拥有者拥有完全控制权。

- 假设在复制配置中，您添加了一个规则以复制具有特定标签的对象子集。在这种情况下，您必须在创建对象时分配特定的标签键和值，以供 Amazon S3 复制对象。如果您先创建对象，然后向现有对象添加标签，Amazon S3 将不会复制对象。
- 可使用 Amazon S3 事件通知，在对象未复制到其目标 AWS 区域的情况下收到通知。Amazon S3 事件通知可以通过 Amazon Simple Queue Service（Amazon SQS）、Amazon Simple Notification Service（Amazon SNS）或 AWS Lambda 获得。有关更多信息，请参阅使用 Amazon S3 事件通知接收复制失败事件。

  您还可以使用 Amazon S3 事件通知查看复制失败原因。要查看失败原因的列表，请参阅 Amazon S3 复制失败原因。

批量复制错误

要对未复制到目标桶的对象进行问题排查，请检查您的桶、复制角色和用于创建批量复制任务的 IAM 角色的不同权限类型。此外，请务必检查公共访问设置和桶所有权设置。

当使用批量复制时，可能会遇到以下错误之一：

- 批量操作状态为失败，原因是：无法将任务报告写入报告桶。

  如果用于批量操作任务的 IAM 角色无法将完成报告放到您创建任务时指定的位置，则会出现此错误。要解决此错误，请检查 IAM 角色是否对您要保存批量操作完成报告的桶具有 `PutObject` 权限。最佳实践是将报告提交到与源桶不同的桶中。
- 批量操作已完成但出现失败，并且失败总数不是 0。

  如果正在运行的批量复制任务存在对象权限不足问题，则会出现此错误。如果您对批量复制任务使用复制规则，请确保用于复制的 IAM 角色具有访问源桶或目标桶中的对象的适当权限。也可以检查批量复制完成报告，以查看特定的 Amazon S3 复制失败原因。
- 批量任务成功运行，但目标桶中预期的对象数量不相同。

  当批量复制任务中提供的清单中列出的对象与您在创建任务时选择的筛选条件不匹配时，就会出现此错误。当源桶中的对象与任何复制规则不匹配且未包含在生成的清单中时，您也可能会收到此消息。

排查服务器访问日志记录问题

以下主题可以帮助您排查在使用 Amazon S3 设置日志记录时可能遇到的问题。

主题

设置日志记录时的常见错误消息

当您通过 AWS Command Line Interface（AWS CLI）和 AWS SDK 启用日志记录功能时，可能会出现以下常见错误消息：

错误：不允许进行跨 S3 位置日志记录

如果目的地存储桶（也称为目标存储桶）与源存储桶位于不同的区域，则会出现不允许跨 S3 位置日志记录错误。要解决此错误，请确保配置为接收访问日志的目标存储桶与源存储桶位于相同的 AWS 区域和 AWS 账户中。

错误：要记录的存储桶的拥有者与目标存储桶的拥有者必须相同

启用服务器访问日志记录时，如果指定的目标存储桶属于其他账户，则会出现此错误。要解决此错误，请确保目标存储桶与源存储桶位于相同 AWS 账户中。

> **Note**
>
> 我们建议您选择不同于源存储桶的目的地存储桶。当源存储桶和目标存储桶相同时，将为写入存储桶的日志创建额外的日志，这会增加您的存储费用。这些关于日志的额外日志也可能使您难以找到您正在寻找的特定日志。为了更方便地管理日志，我们建议将访问日志保存在不同的存储桶中。有关更多信息，请参阅 the section called “如何启用日志传送？”。

错误：用于日志记录的目标存储桶不存在

在设置配置之前，目标存储桶必须已存在。此错误表示目标存储桶不存在或找不到目标存储桶。确保存储桶名称拼写正确，然后重试。

错误：不允许对强制存储桶拥有者的存储存储桶进行目标授权

此错误表明目标存储桶为 S3 对象所有权使用“强制存储桶拥有者”设置。“强制存储桶拥有者”设置不支持目的地（目标）存储桶。有关更多信息，请参阅日志传输的权限。

排查传输失败问题

为避免出现服务器访问日志记录问题，请确保遵循以下最佳实践：

- S3 日志传输组具有对目的地存储桶的写入权限 – S3 日志传输组将服务器访问日志传输到目的地存储桶。存储桶策略或存储桶访问控制列表 (ACL) 可用于授予对目标存储桶的写入权限。但是，我们建议您使用存储桶策略，而不是 ACL。有关如何授予对目标存储桶的写入权限的更多信息，请参阅日志传输的权限。

  > **Note**
  >
  > 如果目标存储桶为对象所有权使用“强制存储桶拥有者”设置，请注意以下几点：
  >
  > - ACL 已禁用，不再影响权限。这意味着，您无法更新存储桶 ACL 以授予对 S3 日志传输组的访问权限。相反，要授予对日志记录服务主体的访问权限，您必须更新目标存储桶的存储桶策略。
  > - 您不能将目标授权包含在 `PutBucketLogging` 配置中。

- 目标存储桶的存储桶策略允许访问日志 – 检查目标存储桶的存储桶策略。在存储桶策略中搜索包含 `"Effect": "Deny"` 的所有语句。然后，验证 `Deny` 语句不会阻止将访问日志写入存储桶中。
- 未对目标存储桶启用 S3 对象锁定 – 检查目标存储桶是否启用了对象锁定。对象锁定阻止服务器访问日志传输。您必须选择未启用对象锁定的目标存储桶。
- 如果在目标存储桶上启用了默认加密，则选择 Amazon S3 托管密钥 (SSE-S3) – 仅当使用具有 Amazon S3 托管式密钥的服务器端加密 (SSE-S3) 时，才能对目标存储桶使用默认存储桶加密。服务器访问日志记录目标存储桶不支持采用 AWS Key Management Service（AWS KMS）密钥的默认服务器端加密 (SSE-KMS)。有关如何启用默认加密的更多信息，请参阅配置默认加密。
- 目标存储桶未启用申请方付款 – 不支持将申请方付款存储桶用作服务器访问日志记录的目标存储桶。要允许传输服务器访问日志，请对目标存储桶禁用申请方付款选项。
- 查看您的 AWS Organizations 服务控制策略 - 当使用 AWS Organizations 时，请检查服务控制策略以确保允许 Amazon S3 访问。服务控制策略指定了受影响账户的最大权限。在服务控制策略中搜索任何包含 `Deny` 的语句，并验证 `"Effect": "Deny"` 语句不会阻止将任何访问日志写入存储桶中。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略（SCP）。
- 等待一段时间让最近的日志记录配置更改生效 – 首次启用服务器访问日志记录或更改日志的目标存储桶时，需要一定的时间才能完全生效。要正确记录和传输所有请求，可能需要一个多小时的时间。

  要检查日志传输失败，请在 Amazon CloudWatch 中启用请求指标。如果日志在几个小时内未传输，请查找 `4xxErrors` 指标，该指标可能表明日志传输失败。有关启用请求指标的更多信息，请参阅the section called “为所有对象创建指标配置”。

排查版本控制问题

以下主题可以帮助您排查一些常见的 Amazon S3 版本控制问题。

主题

我想恢复在启用版本控制的存储桶中意外删除的对象

通常，当从 S3 存储桶中删除对象版本时，Amazon S3 无法恢复它们。但是，如果您在 S3 存储桶上启用了 S3 版本控制，则未指定版本 ID 的 DELETE 请求无法永久删除对象。而是添加删除标记作为占位符。此删除标记将成为对象的当前版本。

要验证已删除的对象是永久删除还是临时删除（在其所在位置有删除标记），请执行以下操作：

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在左侧导航窗格中，选择存储桶。
3. 在存储桶列表中，请选择包含对象的存储桶的名称。
4. 在对象列表中，开启搜索栏右侧的显示版本开关，然后在搜索栏中搜索已删除的对象。仅当之前在存储桶上启用了版本控制时，此开关才可用。

   您也可以使用 S3 清单来搜索已删除的对象。
5. 如果在切换显示版本或创建清单报告后找不到对象，也找不到该对象的删除标记，则删除是永久性的，无法恢复该对象。

您还可以使用 AWS Command Line Interface（AWS CLI）中的 `HeadObject` API 操作来验证已删除对象的状态。要执行此操作，请使用以下 `head-object` 命令并将 *`user input placeholders`* 替换为您自己的信息：

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

如果您对受版本控制的对象运行 `head-object` 命令，而此对象的当前版本为删除标记，则会收到 404 找不到错误。例如：

调用 HeadObject 操作时发生错误（404）：找不到

如果您对受版本控制的对象运行 head-object 命令并提供该对象的版本 ID，Amazon S3 将检索该对象的元数据，确认该对象仍然存在而未被永久删除。

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html --
version-id versionID
```

```
{
"AcceptRanges": "bytes",
"ContentType": "text/html",
"LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
"ContentLength": 77,
"VersionId": "Zg5HyL7m.eZU9iM7AVlJkrqAiE.0UG4q",
"ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
"Metadata": {}
}
```

如果找到该对象并且最新版本是删除标记，则该对象的先前版本仍然存在。由于删除标记是对象的当前版本，因此您可以通过删除该删除标记来恢复对象。

永久移除删除标记后，该对象的第二个最新版本将成为该对象的当前版本，从而使您的对象再次可用。有关如何恢复对象的直观描述，请参阅移除删除标记。

要删除对象的特定版本，您必须是存储桶拥有者。要永久删除“删除标记”，必须在 DeleteObject 请求中包含其版本 ID。要删除该删除标记，请使用以下命令，然后将 *user input placeholders* 替换为您自己的信息：

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key index.html --
version-id versionID
```

有关 delete-object 命令的更多信息，请参阅《AWS CLI 命令参考》中的 delete-object。有关永久删除相应删除标记的更多信息，请参阅管理删除标记。

我想永久删除受版本控制的对象

在已启用版本控制的存储桶中，没有版本 ID 的 DELETE 请求无法永久删除对象。相反，此类请求会插入一个删除标记。

要永久删除受版本控制的对象，可从以下方法中进行选择：

- 创建 S3 生命周期规则以永久删除非当前版本。要永久删除非当前版本，请选择永久删除对象的非当前版本，然后在对象变为非当前对象后的天数下输入一个数字。您可以选择在 Number of newer versions to retain（要保留的较新版本的数量）下输入一个值，从而指定要保留的较新版本数量。有关创建此规则的更多信息，请参阅设置 S3 生命周期配置。
- 通过在 DELETE 请求中包含版本 ID 来删除指定的版本。有关更多信息，请参阅如何永久删除受版本控制的对象。
- 创建生命周期规则以使当前版本过期。要使对象的当前版本过期，请选择将对象的当前版本设为过期，然后在创建对象后的天数下输入一个数字。有关创建此生命周期规则的更多信息，请参阅设置 S3 生命周期配置。
- 要永久删除所有受版本控制的对象以及删除标记，请创建两个生命周期规则：一个用于使当前版本过期并永久删除对象的非当前版本，另一个用于删除过期的对象删除标记。

在启用版本控制的存储桶中，未指定版本 ID 的 DELETE 请求只能删除具有 NULL 版本 ID 的对象。如果对象是在启用版本控制后上传的，则未指定版本 ID 的 DELETE 请求会创建该对象的删除标记。

> **Note**
>
> 对于启用了 S3 对象锁定的存储桶，具有受保护对象版本 ID 的 DELETE 对象请求会导致 403 拒绝访问错误。没有版本 ID 的 DELETE 对象请求会添加删除标记作为该对象的最新版本，响应为 200 OK。受对象锁定保护的对象只有在解除其保留期和法定保留之后才可永久删除。有关更多信息，请参阅 the section called “S3 对象锁定的工作原理”。

启用存储桶版本控制后，我遇到了性能下降问题

如果删除标记或受版本控制的对象过多，以及如果未遵循最佳实践，则启用版本控制的存储桶可能会出现性能降低问题。

删除标记过多

对存储桶启用版本控制后，针对对象发出的没有版本 ID 的 DELETE 请求将创建具有唯一版本 ID 的删除标记。具有将对象的当前版本设为过期规则的生命周期配置会向每个对象添加一个带有唯一版本 ID 的删除标记。过多的删除标记会降低存储桶的性能。

在对存储桶暂停版本控制时，Amazon S3 会对于新创建的对象将版本 ID 标记为 NULL。版本控制已暂停的存储桶中的过期操作会导致 Amazon S3 创建一个以 NULL 作为版本 ID 的删除标记。在版本控制已暂停的存储桶中，将为任何删除请求创建 NULL 删除标记。当删除所有对象版本而只保留单个删除标

记时，这些 NULL 删除标记也称为过期对象删除标记。如果累积的 NULL 删除标记过多，则存储桶的性能会降低。

受版本控制的对象过多

如果启用了版本控制的存储桶包含具有数百万个版本的对象，则可能会出现 503 服务不可用错误。如果您注意到针对启用版本控制的存储存储桶的 PUT 或 DELETE 对象请求接收的 HTTP 503 服务不可用响应的数量显著增加，则存储桶中可能有一个或多个对象具有数百万个版本。如果您的对象有数以百万计的版本，Amazon S3 会自动限制对该存储桶的请求。限制请求数可以保护您的存储桶免受过多请求流量的影响，但也可能会妨碍对该存储桶发出的其他请求。

要确定哪些对象具有数以百万计的版本，请使用 S3 清单。S3 清单生成一份报告，用于提供存储桶中对象的平面文件列表。有关更多信息，请参阅 Amazon S3 清单。

要验证存储桶中是否存在大量受版本控制的对象，请使用 S3 Storage Lens 存储统计管理工具指标来查看当前版本对象计数、非当前版本对象计数和删除标记对象计数。有关 Storage Lens 存储统计管理工具指标的更多信息，请参阅Amazon S3 Storage Lens 存储统计管理工具指标词汇表。

Amazon S3 团队鼓励客户调查重复覆盖同一对象的应用程序（可能会为该对象创建数百万个版本），以确定应用程序是否正常工作。例如，如果某个应用程序每分钟都要覆盖同一个对象，那么在一周时间内创建的版本数量会超过一万个。我们建议，为每个对象存储的版本数不超过十万个。如果您的应用场景需要一个或多个对象的数百万个版本，请联系 AWS Support 团队以获得帮助，从而确定更好的解决方案。

最佳实践

为防止与版本控制相关的性能降级问题，建议您采用以下最佳实践：

- 启用生命周期规则以使对象的先前版本过期。例如，您可以创建生命周期规则，以使非当前版本在对象变为非当前状态的 30 天后过期。如果您不想全部删除非当前版本，也可以保留多个非当前版本。有关更多信息，请参阅设置 S3 生命周期配置。
- 启用生命周期规则，以删除存储桶中没有关联数据对象的过期对象删除标记。有关更多信息，请参阅删除过期对象删除标记。

有关其他 Amazon S3 性能优化最佳实践，请参阅最佳实践设计模式。

获取 AWS Support 的 Amazon S3 请求 ID

每当您因为在 Amazon S3 中遇到错误或意外行为而联系 AWS Support 时，您必须提供与失败操作相关联的请求 ID。AWS Support 使用这些请求 ID 来帮助解决您遇到的问题。

请求 ID 成对出现，并在 Amazon S3 处理的每个响应（甚至是错误的响应）中返回并可通过详细日志访问。可通过许多常见的方法来获取您的请求 ID，包括 S3 访问日志和 AWS CloudTrail 事件或数据事件。

在您恢复这些日志后，复制并保留这两个值，因为您在联系 时需要它们AWS Support 有关联系 AWS Support 的信息，请参阅联系 AWS 或 AWS Support 文档。

使用 HTTP 获得请求 ID

您可以在 HTTP 请求到达目标应用程序之前记录该请求的位数来获取您的请求 ID，即 `x-amz-request-id` 和 `x-amz-id-2`。可使用多种第三方工具来恢复 HTTP 请求的详细日志。选择您信任的某个工具，然后运行该工具，以便在发送出另一个 Amazon S3 HTTP 请求时侦听 Amazon S3 流量通过的端口。

对于 HTTP 请求，请求 ID 对将与以下内容类似：

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

> **Note**
>
> HTTPS 请求将加密并隐藏在大多数数据包捕获中。

使用 Web 浏览器获得请求 ID

大多数 Web 浏览器都具有您可用于查看请求标头的开发人员工具。

对于返回错误的基于 Web 浏览器的请求，请求 ID 对将与以下示例类似。

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOWQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

要从成功的请求中获取请求 ID 对，请使用浏览器的开发人员工具来查看 HTTP 响应标头。有关适用于特定浏览器的开发人员工具的信息，请参阅 AWS re:Post 中的 Amazon S3 疑难解答 – 如何恢复 S3 请求 ID。

使用 AWS SDK 获得请求 ID

以下各节包含有关使用 AWS SDK 配置日志记录的信息。尽管您可以对每个请求和响应启用详细的日志记录，但我们不建议在生产系统中启用日志记录，因为大型请求或响应会显著降低应用程序的速度。

对于 AWS 开发工具包请求，请求 ID 对将与以下示例类似。

```
Status Code: 403, AWS Service: Amazon S3, AWS Request ID: 79104EXAMPLEB723
AWS Error Code: AccessDenied  AWS Error Message: Access Denied
S3 Extended Request ID: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK
+Jd1vEXAMPLEa3Km
```

使用适用于 Go 的 SDK 获取请求 ID

可以使用适用于 Go 的 SDK 配置日志记录。有关更多信息，请参阅《SDK for Go V2 Developer Guide》中的 Response metadata。

使用适用于 PHP 的 SDK 获取请求 ID

可以使用 PHP 配置日志记录。有关更多信息，请参阅《AWS SDK for PHP 开发人员指南》中的如何查看在网上发送了什么数据？

使用适用于 Java 的 SDK 获取请求 ID

可以为特定请求或响应启用日志记录，以仅捕获和返回相关的标头。为此，请导入 `com.amazonaws.services.s3.S3ResponseMetadata` 类。稍后，您可以先将请求存储在变量中，然后执行实际请求。要获取记录的请求或响应，请调用 `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()`。

Example

```
PutObjectRequest req = new PutObjectRequest(bucketName, key, createSampleFile());
s3.putObject(req);
S3ResponseMetadata md = s3.getCachedResponseMetadata(req);
System.out.println("Host ID: " + md.getHostId() + " RequestID: " + md.getRequestId());
```

或者，您可以使用每个 Java 请求和响应的详细日志记录。有关更多信息，请参阅《AWS SDK for Java 开发人员指南》中的详细线路日志记录。

使用 AWS SDK for .NET 获得请求 ID

可以使用内置的 AWS SDK for .NET 日志记录工具通过 `System.Diagnostics` 配置日志记录。有关更多信息，请参阅 AWS 开发人员博客 文章使用适用于 .NET 的 AWS SDK 进行日志记录。

> **Note**
>
> 默认情况下，返回的日志仅包含错误信息。要获取请求 ID，配置文件必须添加了 `AWSLogMetrics`（可以选择添加 `AWSResponseLogging`）。

使用适用于 Python 的 SDK（Boto3）获取请求 ID

使用 AWS SDK for Python (Boto3)，您可以记录特定的响应。您可以使用此功能仅捕获相关的标题。以下代码显示如何将响应的各个部分记录到文件中：

```
import logging
import boto3
logging.basicConfig(filename='logfile.txt', level=logging.INFO)
logger = logging.getLogger(__name__)
s3 = boto3.resource('s3')
response = s3.Bucket(bucket_name).Object(object_key).put()
logger.info("HTTPStatusCode: %s", response['ResponseMetadata']['HTTPStatusCode'])
logger.info("RequestId: %s", response['ResponseMetadata']['RequestId'])
logger.info("HostId: %s", response['ResponseMetadata']['HostId'])
logger.info("Date: %s", response['ResponseMetadata']['HTTPHeaders']['date'])
```

还可以在引发异常时捕获异常并记录相关信息。有关更多信息，请参阅《适用于 Python 的 AWS SDK（Boto3）API 参考》中的从错误响应中识别有用信息。

此外，您可以使用以下代码将 Boto3 配置为输出详细调试日志：

```
import boto3
boto3.set_stream_logger('', logging.DEBUG)
```

有关更多信息，请参阅《适用于 Python 的 AWS SDK（Boto3）API 参考》中的 set_stream_logger。

使用适用于 Ruby 的 SDK 获取请求 ID

您可以使用适用于 Ruby 的 SDK 版本 1、2 或 3 来获取请求 ID。

- 使用适用于 Ruby 的开发工具包 - 版本 1 – 您可以使用以下代码行来全局启用 HTTP 线路日志记录。

```
s3 = AWS::S3.new(:logger => Logger.new($stdout), :http_wire_trace => true)
```

- 使用适用于 Ruby 的开发工具包 - 版本 2 或版本 3 – 您可以使用以下代码行来全局启用 HTTP 线路日志记录。

```
s3 = Aws::S3::Client.new(:logger => Logger.new($stdout), :http_wire_trace => true)
```

有关从 AWS 客户端获取线路信息的提示，请参阅调试提示：从客户端获取线路跟踪信息。

使用 AWS CLI 获得请求 ID

要在使用 AWS Command Line Interface（AWS CLI）时获取您的请求 ID，请将 `--debug` 添加到您的命令中。

使用 Windows PowerShell 获取请求 ID

有关使用 Windows PowerShell 恢复日志的信息，请参阅 .NET 开发博客文章 AWS Tools for Windows PowerShell 中的响应日志记录。

使用 AWS CloudTrail 数据事件获取请求 ID

使用 CloudTrail 数据事件配置为记录 S3 对象级别 API 操作的 Amazon S3 桶提供了有关用户、角色或 AWS 服务在 Amazon S3 桶中所采取操作的详细信息。您可以通过向 Athena 查询 CloudTrail 事件来识别 S3 请求 ID。

使用 S3 服务器访问日志记录获取请求 ID

为 S3 服务器访问日志记录配置的 Amazon S3 桶为向桶发出的每个请求提供详细记录。您可以通过使用 Athena 查询服务器访问日志来识别 S3 请求 ID。

文档历史记录

- 当前 API 版本：2006-03-01

下表介绍了 Amazon Simple Storage Service API 参考和 Amazon S3 用户指南的每个版本中的重要更改。要获得本文档的更新通知，您可以订阅 RSS 源。

变更	说明	日期
Amazon S3 支持对 PutObject 和 CompleteMultipartUpload 使用有条件写入	在对上传操作使用有条件写入来创建对象之前，您可以检查存储桶中是否存在该对象。这样可以防止覆盖现有数据。有条件写入将验证存储桶中尚不存在具有相同键名称的现有对象。有关更多信息，请参阅 Conditional requests。	2024 年 8 月 20 日
不再向新客户提供 Amazon S3 Select	不再向新客户提供 Amazon S3 Select。Amazon S3 Select 的现有客户可以像往常一样继续使用该功能。了解更多。	2024 年 7 月 25 日
Amazon S3 清单支持 s3:InventoryAccessibleOptionalFields 条件键	Amazon S3 清单支持 s3:InventoryAccessibleOptionalFields 条件键，以便控制用户能否在其报告中包含可选的元数据字段。有关更多信息，请参阅控制 S3 清单报告配置创建。	2024 年 2 月 20 日
S3 on Outposts 的 IPv6 支持	现在，可以通过 S3 on Outposts 双堆栈端点，使用 IPv6 访问 S3 on Outposts 桶。S3 on Outposts 的 IPv6 支持可让您通过 IPv6 网络管理	2024 年 1 月 16 日

	S3 on Outposts 桶和控制面板资源。	
全新的高性能、单区 Amazon S3 存储类 – S3 Express One Zone	Amazon S3 Express One Zone 是高性能的单区 Amazon S3 存储类，专门用于为延迟要求极高的应用程序提供稳定的毫秒级数据访问。有关更多信息，请参阅 S3 Express One Zone。	2023 年 11 月 28 日
适用于 Amazon S3 的 Mountpoint 增加了对 S3 Express One Zone 的支持	现在可以使用 Mountpoint 挂载 S3 Express One Zone 目录存储桶。	2023 年 11 月 28 日
Lambda 调用架构版本	Amazon S3 批量操作引入了新的 Lambda 调用架构版本，用于对目录存储桶执行的批量操作作业。有关更多信息，请参阅将 Lambda 和 Amazon S3 批量操作用于目录存储桶。	2023 年 11 月 28 日
目录存储桶的导入操作	Amazon S3 引入了导入操作。导入是一种创建 Amazon S3 批量操作作业的简化方法，用于将对象从通用存储桶复制到目录存储桶。有关更多信息，请参阅将对象导入到目录存储桶。	2023 年 11 月 28 日

使用 S3 Access Grants 管理 S3 访问权限	Amazon S3 Access Grants 使您能够大规模地管理 AWS Identity and Access Management（IAM）主体的数据权限，以及来自企业目录（如 Azure AD）的目录标识。现在，您可以强制执行最低权限的 S3 权限，并根据业务需求轻松扩展这些权限。有关更多信息，请参阅使用 S3 Access Grants 管理访问权限。	2023 年 11 月 26 日
适用于 Amazon S3 的 Mountpoint 增加了缓存功能	借助 Mountpoint，您现在可以为重复访问的数据配置缓存。	2023 年 11 月 22 日
增强 Amazon S3 批量操作清单生成	现在，您可以指示 Amazon S3 批量操作根据创建作业时指定的对象筛选条件自动生成清单。此选项适用于您在 Amazon S3 控制台中创建的批量复制任务，也适用于您使用 AWS CLI、AWS SDK 或 Amazon S3 REST API 创建的任何任务类型。有关更多信息，请参阅创建 Amazon S3 批量操作作业。	2023 年 11 月 22 日
现有的 Amazon S3 存储桶现在可以添加对象锁定配置	现在，您可以对现有的 Amazon S3 存储桶启用对象锁定。您可以为新的或现有的存储桶设置依法持有和保留期。有关更多信息，请参阅使用对象锁定。	2023 年 11 月 20 日

S3 Storage Lens 存储统计管理工具前缀请求指标	S3 Storage Lens 存储统计管理工具在 Amazon S3 存储桶中引入了前缀请求指标。有关更多信息，请参阅指标类别。	2023 年 11 月 17 日
Amazon S3 Storage Lens 存储统计管理工具组	S3 Storage Lens 存储统计管理工具引入了 Storage Lens 组，这是一种基于对象元数据的自定义对象筛选条件。有关更多信息，请参阅 Working with Amazon S3 Storage Lens groups。	2023 年 11 月 15 日
新的 IAM 策略	S3 on Outposts 引入了 `AWSServiceRoleForS3OnOutposts` ，这是一个服务相关角色，可帮助您管理网络资源。有关更多信息，请参阅使用面向 S3 on Outposts 的服务相关角色。	2023 年 10 月 3 日
Amazon S3 提供了删除标记的 `Last-Modified` 时间	Amazon S3 在 S3 `Head` 和 `Get` API 操作的响应标头中提供了删除标记的 `Last-Modified` 时间。有关更多信息，请参阅使用删除标记。	2023 年 9 月 27 日
AWS 托管式策略的 Amazon S3 更新	Amazon S3 向 `AmazonS3ReadOnlyAccess` 添加了 `s3:Describe*` 权限。有关更多信息，请参阅 Amazon S3 的 AWS 托管式策略。	2023 年 8 月 11 日
缩短了通过 S3 批量操作发出的标准还原请求的启动时间	现在，通过 S3 批量操作发出的还原请求的标准检索可以在几分钟内启动。有关更多信息，请参阅归档检索选项。	2023 年 8 月 9 日

添加了 Mountpoint，这是一款用于将 Amazon S3 存储桶作为本地文件系统挂载的高吞吐量客户端。	借助 Mountpoint，您的应用程序可以通过文件操作访问存储在 Amazon S3 中的对象，同时使您的应用程序能够通过文件接口访问 Amazon S3 的弹性存储和吞吐量。	2023 年 8 月 9 日
具有 AWS Key Management Service 密钥的双层服务器端加密（DSSE-KMS）	在将对象上传到 Amazon S3 时，具有 AWS Key Management Service（AWS KMS）密钥的双层服务器端加密（DSSE-KMS）将会对于对象应用两层加密。有关更多信息，请参阅使用具有 AWS KMS 密钥的双层服务器端加密。	2023 年 6 月 13 日
Amazon S3 为所有新存储桶启用 S3 屏蔽公共访问权限并禁用 S3 访问控制列表（ACL）。	Amazon S3 现在会自动启用 S3 屏蔽公共访问权限，并对所有 AWS 区域中的所有新 S3 存储桶禁用 S3 访问控制列表（ACL）。有关更多信息，请参阅屏蔽 Amazon S3 存储的公共访问权限和控制对象所有权和禁用存储桶的 ACL。	2023 年 4 月 27 日
S3 复制操作失败指标	Amazon S3 添加了新的 Amazon CloudWatch 指标来监控 S3 复制失败。有关更多信息，请参阅使用复制指标监控进度。	2023 年 4 月 5 日
私有 DNS	适用于 Amazon S3 的 AWS PrivateLink 现在支持私有 DNS。有关更多信息，请参阅私有 DNS。	2023 年 3 月 14 日

Amazon S3 控制台中的跨账户接入点支持	Amazon S3 现在支持使用 Amazon S3 控制台创建跨账户接入点。有关更多信息，请参阅创建接入点。	2023 年 3 月 14 日
Amazon S3 on Outposts 支持 Outposts 上的 S3 复制	使用本地 S3 复制，您可以自动将对象复制到单个 Outposts 目标存储桶或多个目标存储桶。目标存储桶可以位于不同的 AWS Outposts，也可以与源存储桶位于同一 Outposts 中。有关更多信息，请参阅复制 S3 on Outposts 的对象。	2023 年 3 月 14 日
Amazon S3 对象 Lambda 接入点别名	当您创建对象 Lambda 接入点时，Amazon S3 会自动为您的对象 Lambda 接入点生成一个唯一的别名。您可以在接入点数据面板操作请求中使用此别名，而不使用 Amazon S3 存储桶名称或对象 Lambda 接入点 Amazon 资源名称（ARN）。有关更多信息，请参阅如何为您的对象 Lambda 接入点使用存储桶式别名。	2023 年 3 月 14 日
Amazon S3 多区域接入点跨账户支持	Amazon S3 现在支持使用 Amazon S3 控制台创建跨账户多区域接入点。有关更多信息，请参阅创建多区域接入点。	2023 年 3 月 14 日

跨账户接入点	Amazon S3 支持创建跨账户接入点。您可以使用 AWS Command Line Interface（AWS CLI）或 REST API `CreateAccessPoint` 操作创建跨账户接入点。有关更多信息，请参阅创建接入点。	2022 年 11 月 30 日
Amazon S3 支持对 Amazon S3 多区域接入点进行失效转移控制	Amazon S3 为多区域接入点引入了失效转移控制。这些控制措施允许您在几分钟内将通过 Amazon S3 多区域接入点路由的 S3 数据访问请求流量转移到备用 AWS 区域，以测试和构建高可用性应用程序。有关更多信息，请参阅 Amazon S3 多区域接入点失效转移控制。	2022 年 11 月 28 日
Amazon S3 Storage Lens 存储统计管理工具通过 34 个新指标提高了整个组织的可见性	S3 Storage Lens 存储统计管理工具引入了 34 个附加指标，以发现更深的成本优化机会，确定数据保护最佳实践，并提高应用程序工作流的性能。有关更多信息，请参阅 S3 Storage Lens 存储统计管理工具指标。	2022 年 11 月 17 日
Amazon S3 对于 S3 Glacier Flexible Retrieval 和 S3 Glacir Deep Archive 支持更高的还原请求速率	对于 S3 Glacier Flexible Retrieval 和 S3 Glacier Deep Archive 存储类，Amazon S3 支持以每个 AWS 账户每秒最多 1000 个事务的速率发出还原请求。	2022 年 11 月 15 日

Amazon S3 on Outposts 支持其他 S3 生命周期操作和筛选条件	S3 on Outposts 支持其他 S3 生命周期规则，以优化容量管理。您可以随着对象的老化而使它们过期，也可以将其替换为较新的版本。通过使用前缀、对象标签或对象大小进行筛选，可以为整个存储桶或存储桶中的对象子集创建生命周期规则。有关更多信息，请参阅创建和管理生命周期配置。	2022 年 11 月 2 日
对 SSE-C 对象的 S3 复制支持	您可以复制使用具有客户提供的加密密钥的服务器端加密创建的对象。有关复制加密对象的更多信息，请参阅复制使用服务器端加密（SSE-C、SSE-S3、SSE-KMS）创建的对象。	2022 年 10 月 24 日
Amazon S3 on Outposts 支持接入点别名	对于 S3 on Outposts，您必须使用接入点访问 Outposts 存储桶中的任何对象。每次您为存储桶创建接入点时，S3 on Outposts 都会自动生成一个接入点别名。对于任何数据面板操作，您都可以使用此接入点别名，而不是接入点 ARN。有关更多信息，请参阅为 S3 on Outposts 存储桶接入点使用存储桶式别名。	2022 年 10 月 21 日
S3 对象 Lambda 支持 `HeadObject` 、`ListObjects` 和 `ListObjectsV2` 操作	您可以使用自定义代码修改标准 S3 GET、LIST 或 HEAD 请求返回的数据，以便执行筛选行、动态调整图像大小、隐去机密数据等操作。有关更多信息，请参阅使用 S3 对象 Lambda 转换对象。	2022 年 10 月 4 日

Amazon S3 on Outposts 支持 S3 版本控制	启用后，S3 版本控制功能将对象的多个不同副本保存到同一个存储桶中。对于 Outpost 桶中存储的每个对象，您可以使用 S3 版本控制功能来保留、检索和还原其每个版本。S3 版本控制功能可帮助您从用户意外操作和应用程序故障中恢复。有关更多信息，请参阅为 S3 on Outposts 存储桶管理 S3 版本控制。	2022 年 9 月 21 日
适用于 Amazon S3 的 AWS Backup	AWS Backup 是一种完全托管式的、基于策略的服务，您可以使用它来定义中央备份策略，以保护您的 Amazon S3 数据。有关更多信息，请参阅对 Amazon S3 使用 AWS Backup。	2022 年 2 月 18 日
使用 S3 分批复制以复制现有对象	使用 S3 分批复制，您可以复制在实施复制配置之前就已存在的对象。复制现有对象是通过使用批量操作任务完成的。S3 分批复制与实时复制不同，后者连续自动跨 Amazon S3 存储桶复制新对象。有关更多信息，请参阅使用 S3 批量复制来复制现有对象。	2022 年 2 月 8 日
重命名 S3 Glacier Flexible Retrieval	Glacier 存储类已重命名为 S3 Glacier Flexible Retrieval。此更改不影响 API。	2021 年 11 月 30 日

用于禁用 ACL 的新 S3 对象所有权设置	您可以应用对象拥有权的强制存储桶拥有者设置，以对存储桶及其中的对象禁用 ACL，并获取存储桶中每个对象的拥有权。强制存储桶拥有者设置简化了对存储在 Amazon S3 中的数据的访问管理。有关更多信息，请参阅 Controlling ownership of objects and disabling ACLs for your bucket（为您的存储桶控制对象所有权和禁用 ACL）。	2021 年 11 月 30 日
新建 S3 Intelligent-Tiering 存储类	S3 Intelligent-Tiering 归档即时访问是 S3 Intelligent-Tiering 下的额外存储类。有关更多信息，请参阅 S3 Intelligent-Tiering 的工作原理。	2021 年 11 月 30 日
新的 S3 Glacier 即时检索存储类	您现在可以将对象放在 S3 Glacier 即时检索存储类中。有关此存储类的更多信息，请参阅使用 Amazon S3 存储类。	2021 年 11 月 30 日
适用于 Amazon S3 的 AWS Backup 预览版	AWS Backup 是一种完全托管式的、基于策略的服务，您可以使用它来定义中央备份策略，以保护您的 Amazon S3 数据。有关更多信息，请参阅对 Amazon S3 使用 AWS Backup。	2021 年 11 月 30 日

适用于 Amazon S3 的 AWS Identity and Access Management Access Analyzer	IAM Access Analyzer 将根据 IAM 策略语法和最佳实践运行策略检查，以验证您的策略。要了解有关使用 IAM Access Analyzer 验证策略的更多信息，请参阅《IAM 用户指南》中的 IAM Access Analyzer 策略验证。	2021 年 11 月 30 日
新事件类型	Amazon S3 事件通知添加了新事件类型，请参阅 Amazon S3 Event Notifications（Amazon S3 事件通知）。	2021 年 11 月 29 日
在存储桶上启用 Amazon EventBridge	您可以在 Amazon S3 存储桶上启用 EventBridge 以将事件发送到 Amazon EventBridge，请参阅使用 EventBridge。	2021 年 11 月 29 日
新 S3 生命周期筛选器	您可以根据对象大小创建生命周期规则，也可以指定要保留多少个非当前对象版本。有关更多信息，请参阅 S3 生命周期配置示例。	2021 年 11 月 23 日

将 Amazon S3 Storage Lens 存储统计管理工具指标发布到 Amazon CloudWatch	您可以将 S3 Storage Lens 存储统计管理工具使用情况和活动指标发布到 Amazon CloudWatch，以便在 CloudWatch 控制面板中创建运营状况的统一视图。您还可以使用 CloudWatch 特征（如警报和触发操作、指标数学和异常检测）来监控 S3 Storage Lens 存储统计管理工具指标并采取措施。此外，CloudWatch API 使应用程序（包括第三方提供商）能够访问 S3 Storage Lens 存储统计管理工具指标。有关更多信息，请参阅 Monitor S3 Storage Lens metrics in CloudWatch（在 CloudWatch 中监控 S3 Storage Lens 存储统计管理工具指标）。	2021 年 11 月 22 日
多区域接入点	您可以使用多区域接入点创建全局端点，应用程序可以使用该端点来满足来自位于多个 AWS 区域的 Amazon S3 存储桶的请求。您可以使用此多区域接入点将数据路由到具有最低延迟的存储桶。有关多区域接入点及其使用方法的更多信息，请参阅 Amazon S3 中的多区域接入点。	2021 年 9 月 2 日

Amazon S3 on Outposts 为应用程序添加直接本地访问	从 AWS Outposts Virtual Private Cloud（VPC）外部运行应用程序，并访问 S3 on Outposts 数据。您还可以直接从您的本地网络访问 S3 on Outposts 对象。有关使用客户拥有的 IP（CoIP）地址配置 S3 on Outposts 端点，以及通过从本地网络创建本地网关访问对象的详细信息，请参阅使用仅限 VPC 接入点访问 Amazon S3 on Outposts。	2021 年 7 月 29 日
Amazon S3 接入点别名	当您创建接入点时，Amazon S3 会自动生成一个别名。您可以使用该别名来替代存储桶名称以进行数据访问。对于任何接入点数据层面操作，您都可以使用此接入点别名而不是 Amazon Resource Name（ARN）。有关更多信息，请参阅为您的接入点使用存储桶式别名。	2021 年 7 月 26 日
Amazon S3 清单和 S3 批量操作支持 S3 存储桶密钥状态	Amazon S3 清单和批量操作支持使用 S3 存储桶密钥识别和复制现有对象。S3 存储桶密钥可加快降低对现有对象进行服务器端加密的成本。有关更多信息，请参阅 Amazon S3 清单和批量操作复制对象。	2021 年 6 月 3 日

Amazon S3 Storage Lens 存储统计管理工具指标账户快照	S3 Storage Lens 存储统计管理工具账户快照通过总结默认控制面板中的指标，在 S3 控制台主页（存储桶）上显示您的总存储空间、对象计数和平均对象大小。有关更多信息，请参阅 S3 Storage Lens 存储统计管理工具指标账户快照。	2021 年 5 月 5 日
增加了 Amazon S3 on Outposts 端点支持	S3 on Outposts 现在支持在每个 Outposts 多达 100 个端点。有关更多信息，请参阅 S3 on Outposts 网络限制。	2021 年 4 月 29 日
Amazon CloudWatch Events 中的 Amazon S3 on Outposts 事件通知	您现在可以使用 CloudWatch Events 捕获任意 S3 on Outposts API 事件，并通过所有受支持的 CloudWatch 目标获取通知。有关更多信息，请参阅使用 CloudWatch Events 接收 S3 on Outposts 事件通知。	2021 年 4 月 19 日
S3 对象 Lambda	借助 S3 对象 Lambda，您可以将自己的代码添加到 Amazon S3 GET 请求中，以便在数据返回到应用程序时修改和处理数据。您可以使用自定义代码来修改标准 S3 GET 请求返回的数据，以便执行筛选行、动态调整图像大小、隐去机密数据等操作。有关更多信息，请参阅转换对象。	2021 年 3 月 18 日

AWS PrivateLink	借助适用于 Amazon S3 的 AWS PrivateLink，您可以使用虚拟私有云（VPC）中的接口端点直接连接到 S3，而不是通过互联网进行连接。接口端点可从本地或其他 AWS 区域中的应用程序直接访问。有关更多信息，请参阅适用于 Amazon S3 的 AWS PrivateLink。	2021 年 2 月 2 日
使用 AWS CloudTrail 管理 Amazon S3 on Outposts 容量	S3 on Outposts 管理事件可通过 CloudTrail 日志提供。有关更多信息，请参阅使用 CloudTrail 管理 S3 on Outposts 容量。	2020 年 12 月 21 日
强一致性	Amazon S3 为针对所有 PUT的 S3 存储桶中对象的 DELETE 和 AWS 区域 请求提供了强大的先写后读一致性。此外，针对 Amazon S3 Select、Amazon S3 访问控制列表、Amazon S3 对象标签和对象元数据（例如 HEAD 对象）的读取操作具有严格的一致性。有关更多信息，请参阅 Amazon S3 数据一致性模型。	2020 年 12 月 1 日
Amazon S3 副本修改同步	Amazon S3 副本修改同步使对象元数据（如标签、ACL 和对象锁定设置）在源对象和副本之间保持同步。启用此特征后，Amazon S3 会复制对源对象或副本所做的元数据更改。有关更多信息，请参阅使用副本修改同步复制元数据更改。	2020 年 12 月 1 日

Amazon S3 存储桶密钥	Amazon S3 存储桶密钥降低了具有 AWS Key Management Service 的 Amazon S3 服务器端加密（SSE-KMS）的成本。这一用于服务器端加密的存储桶级别密钥通过减少从 Amazon S3 到 AWS KMS 的请求流量，将 AWS KMS 请求成本降低多达 99%。有关更多信息，请参阅使用 S3 存储桶密钥降低 SSE-KMS 的成本。	2020 年 12 月 1 日
Amazon S3 Storage Lens 存储统计管理工具	S3 Storage Lens 存储统计管理工具聚合您的指标，并在 Amazon S3 控制台的 Buckets（存储桶）页上的 Account snapshot（账户快照）部分中显示此信息。S3 Storage Lens 存储统计管理工具还提供了一个交互式控制面板，您可以使用它来可视化见解和趋势，标记异常值，并接收有关优化存储成本和应用数据保护最佳实践的建议。控制面板提供深入分析选项，用于在组织、账户、AWS 区域、存储类、存储桶、前缀或 Storage Lens 组级别生成和可视化见解。您还可以将采用 CSV 或 Parquet 格式的每日指标导出发送到 S3 存储桶。有关更多信息，请参阅使用 S3 Storage Lens 存储统计管理工具访问存储活动和使用情况。	2020 年 11 月 18 日

使用 AWS X-Ray 跟踪 S3 请求	Amazon S3 与 X-Ray 集成，以传播跟踪上下文，并为您提供一个带有上游和下游节点的请求链。有关更多信息，请参阅使用 X-Ray 跟踪请求。	2020 年 11 月 16 日
S3 复制指标	S3 复制指标为复制配置中的复制规则提供了详细的指标。有关更多信息，请参阅复制指标和 Amazon S3 事件通知。	2020 年 11 月 9 日
S3 Intelligent-Tiering 存档访问和深度存档访问	S3 Intelligent-Tiering 存档访问和深度存档访问是 S3 Intelligent-Tiering 下的额外存储层。有关更多信息，请参阅可自动优化经常访问和不常访问对象的存储类。	2020 年 11 月 9 日
删除标记复制	通过删除标记复制，您可以确保将删除标记复制到复制规则的目标存储桶中。有关更多信息，请参阅使用删除标记复制。	2020 年 11 月 9 日
S3 对象所有权	对象所有权是一个 S3 存储桶设置，您可以使用它来控制上传到存储桶的新对象的所有权。有关更多信息，请参阅使用 S3 对象所有权。	2020 年 10 月 2 日

Amazon S3 on Outposts	通过使用 Amazon S3 on Outposts，您可以在 AWS Outposts 资源上创建 S3 存储桶，并在本地为需要本地数据访问、本地数据处理和数据驻留的应用程序轻松地存储和检索对象。您可以通过 AWS Management Console、AWS CLI、AWS SDK 或 REST API 使用 S3 on Outposts。有关更多信息，请参阅使用 Amazon S3 on Outposts。	2020 年 9 月 30 日
存储桶拥有者条件	您可以使用 Amazon S3 存储桶拥有者条件来确保您在 S3 操作中使用的存储桶属于您期望的 AWS 账户。有关更多信息，请参阅存储桶拥有者条件。	2020 年 9 月 11 日
S3 批量操作支持对象锁定保留	现在，您可以将批量操作与 S3 对象锁定一起使用，以便同时对许多 Amazon S3 对象应用保留设置。有关更多信息，请参阅使用 S3 批量操作设置 S3 对象锁定保留日期。	2020 年 5 月 4 日
S3 批量操作支持对象锁定依法保留	现在，您可以将批量操作与 S3 对象锁定一起使用，以便同时为许多 Amazon S3 对象添加依法保留。有关更多信息，请参阅使用 S3 批量操作设置 S3 对象锁定依法保留。	2020 年 5 月 4 日
S3 批量操作的任务标签	您可以向 S3 批量操作作业添加标签以控制和标记这些作业。有关更多信息，请参阅 S3 批量操作任务的标签。	2020 年 3 月 16 日

Amazon S3 接入点	Amazon S3 接入点可简化对 S3 中的共享数据集的大规模数据访问管理。接入点是附加到存储桶的具名网络端点，您可以使用这些存储桶执行 S3 对象操作。有关更多信息，请参阅使用 Amazon S3 接入点管理数据访问。	2019 年 12 月 2 日
Amazon S3 的访问分析器	如果存在已配置为允许 Internet 上的任何人或其他 AWS 账户（包括组织外部的账户）访问的 S3 存储桶，Amazon S3 访问分析器会向您发出提醒。有关更多信息，请参阅使用 Amazon S3 的访问分析器。	2019 年 12 月 2 日
S3 复制时间控制 (S3 RTC)	S3 复制时间控制 (S3 RTC) 可以在几秒钟内复制您上传到 Amazon S3 的大部分对象，其中 99.99% 的对象会在 15 分钟内完成复制。有关更多信息，请参阅使用 S3 Replication Time Control（S3 RTC）复制对象。	2019 年 11 月 20 日
同区域复制	可以使用同区域复制（SRR）在同一 AWS 区域中跨 Amazon S3 存储桶复制对象。有关跨区域复制（CRR）和同区域复制的信息，请参阅复制。	2019 年 9 月 18 日
S3 对象锁定的跨区域复制支持	跨区域复制现在支持对象锁定。有关更多信息，请参阅 Amazon S3 会复制什么？	2019 年 5 月 28 日

S3 批量操作	使用 S3 批量操作，您可以对 Amazon S3 对象执行大规模批量操作。S3 批量操作可以对您指定的对象列表运行单个操作。单个作业可对包含 EB 级数据的数十亿个对象执行指定操作。有关更多信息，请参阅执行 S3 批处理操作。	2019 年 4 月 30 日
亚太地区（香港）区域	Amazon S3 现已在亚太地区（香港）区域中提供。有关 Amazon S3 区域和端点的更多信息，请参阅《AWS 一般参考》中的区域和端点。	2019 年 4 月 24 日
向服务器访问日志添加了新字段	Amazon S3 向服务器访问日志添加了以下新字段：传输层安全性 (TLS) 版本。有关更多信息，请参阅服务器访问日志格式。	2019 年 3 月 28 日
新归档存储类	Amazon S3 现在提供了用于存储极少访问的对象的新归档存储类 S3 Glacier Deep Archive（DEEP_ARCHIVE ）。有关更多信息，请参阅存储类。	2019 年 3 月 27 日
向服务器访问日志添加了新字段	Amazon S3 向服务器访问日志添加了以下新字段：主机 ID、签名版本、密码包、身份验证类型和主机标头。有关更多信息，请参阅服务器访问日志格式。	2019 年 3 月 5 日

支持 Parquet 格式的 Amazon S3 清单文件	除了 Apache optimized row columnar（ORC）和逗号分离值（CSV）文件格式之外，Amazon S3 现在还支持 Apache Parquet（Parquet）格式。有关更多信息，请参阅清单。	2018 年 12 月 4 日
S3 对象锁定	Amazon S3 现在提供可为 Amazon S3 对象提供一次写入、多次读取（WORM）保护的对象锁定功能。有关更多信息，请参阅锁定对象。	2018 年 11 月 26 日
还原速度升级	使用 Amazon S3 还原速度升级，您可以在从 S3 Glacier Flexible Retrieval 存储类进行还原的过程中将该还原的速度更改为更快的速度。有关更多信息，请参阅还原归档对象。	2018 年 11 月 26 日
还原事件通知	现在，Amazon S3 事件通知支持在从 S3 Glacier Flexible Retrieval 存储类还原对象时的启动和完成事件。有关更多信息，请参阅事件通知。	2018 年 11 月 26 日

直接对 S3 Glacier Flexible Retrieval 存储类执行 PUT 操作	现在，Amazon S3 PUT 操作支持在创建对象时将 S3 Glacier Flexible Retrieval 指定为存储类。以前，您必须将对象从其他 Amazon S3 存储类转换为 S3 Glacier Flexible Retrieval 存储类。而且，在使用 S3 跨区域复制 (CRR) 时，现在您可以将 S3 Glacier Flexible Retrieval 指定为已复制对象的存储类。有关 S3 Glacier Flexible Retrieval 存储类的更多信息，请参阅存储类。有关为已复制的对象指定存储类的更多信息，请参阅复制配置概述。有关直接对 S3 Glacier Flexible Retrieval REST API 执行 PUT 操作这一更改的更多信息，请参阅文档历史记录：直接对 S3 Glacier Flexible Retrieval 执行 PUT 操作。	2018 年 11 月 26 日
新存储类	Amazon S3 现在提供一种名为 S3 Intelligent-Tierin g（`INTELLIGENT_TIERIN G`）的新存储类，这是专为不断变化或访问模式未知的长期数据设计的。有关更多信息，请参阅存储类。	2018 年 11 月 26 日
Amazon S3 屏蔽公共访问权限	Amazon S3 现在包含按存储桶或账户对存储桶和对象屏蔽公共访问权限的功能。有关更多信息，请参阅使用 Amazon S3 屏蔽公共访问权限。	2018 年 11 月 15 日

跨区域复制 (CRR) 规则中的筛选增强功能	在 CRR 规则配置中，您可以指定对象筛选条件以选择要将向其应用规则的对象子级。以前，您只能按对象键前缀进行筛选。在本版本中，您可以按对象键前缀和/或一个或多个对象标签进行筛选。有关更多信息，请参阅 CRR 设置：复制配置概述。	2018 年 9 月 19 日
新的 Amazon S3 Select 特征	Amazon S3 Select 现在支持 Apache Parquet 输入、对嵌套 JSON 对象的查询以及两个新的 Amazon CloudWatch 监控指标（`SelectScannedBytes` 和 `SelectReturnedBytes`）。	2018 年 9 月 5 日
现在可通过 RSS 更新	您现在可以订阅 RSS 源来接收有关《Amazon S3 用户指南》更新的通知。	2018 年 19 月 6 日

早期更新

下表描述了 2018 年 6 月 19 日之前每次发布 Amazon S3 用户指南时进行的重要更改。

更改	描述	日期
代码示例更新	更新了代码示例： • C# — 已将所有示例更新为使用基于任务的异步模式。有关更多信息，请参阅《AWS SDK for .NET 开发人员指南》中的适用于 .NET 的 Amazon Web Services 异步 API。代码示例现在符合版本 3 的 AWS SDK for .NET 的要求。 •	2018 年 4 月 30 日

更改	描述	日期
	Java — 已将所有示例更新为使用客户端生成器模型。有关客户端生成器模型的更多信息，请参阅创建服务客户端。 • PHP – 已将所有示例更新为使用 AWS SDK for PHP 3.0。有关 AWS SDK for PHP 3.0 的更多信息，请参阅 AWS SDK for PHP。 • Ruby – 已更新示例代码，以便示例可以与 AWS SDK for Ruby 版本 3 结合使用。	
Amazon S3 现在将 S3 Glacier Flexible Retrieval 和 ONEZONE_IA 存储类报告给 Amazon CloudWatch Logs 存储指标	除了报告实际字节数之外，这些存储指标还包含合适的存储类（ONEZONE_IA 、STANDARD_IA 和 S3 Glacier Flexible Retrieval）的每对象开销字节数： • 对于 ONEZONE_IA 和 STANDARD_IA 存储类对象，Amazon S3 将小于 128 KB 的对象报告为 128 KB。有关更多信息，请参阅 使用 Amazon S3 存储类。 • 对于 S3 Glacier Flexible Retrieval 存储类对象，存储指标将报告以下开销： • 一个 32 KB 每对象开销，按 S3 Glacier Flexible Retrieval 存储类定价收费 • 8KB 每对象开销，按 STANDARD 存储类定价收费 有关更多信息，请参阅 使用 Amazon S3 生命周期转换对象。 有关存储指标的更多信息，请参阅 使用 Amazon CloudWatch 监控指标。	2018 年 30 月 4 日

更改	描述	日期
新存储类	Amazon S3 现在提供了用于存储对象的新存储类 `STANDARD_IA` （IA，适用于不频繁访问）。此存储类已针对长时间运行且不常访问的数据进行了优化。有关更多信息，请参阅 使用 Amazon S3 存储类。	2018 年 4 月 4 日
Amazon S3 Select	Amazon S3 现在支持根据 SQL 表达式检索对象内容。有关更多信息，请参阅 使用 Amazon S3 Select 筛选和检索数据。	2018 年 4 月 4 日
亚太区域 (大阪当地)	Amazon S3 现已在亚太地区（大阪本地）区域中提供。有关 Amazon S3 区域和端点的更多信息，请参阅《AWS 一般参考》中的区域和端点。 **Important** 您只能与亚太地区（东京）区域结合使用亚太地区（大阪本地）区域。要请求访问亚太地区（大阪本地）区域，请联系您的销售代表。	2018 年 2 月 12 日
Amazon S3 清单创建时间戳	Amazon S3 清单现在包括创建 Amazon S3 清单报告的日期和开始时间的时间戳。您可以使用此时间戳确定开始生成清单报告的时间之后 Amazon S3 存储中的变化。	2018 年 1 月 16 日
欧洲（巴黎）区域	Amazon S3 现已在欧洲地区（巴黎）区域中提供。有关 Amazon S3 区域和端点的更多信息，请参阅《AWS 一般参考》中的区域和端点。	2017 年 12 月 18 日
中国 (宁夏) 区域	Amazon S3 现已在中国（宁夏）区域中提供。有关 Amazon S3 区域和端点的更多信息，请参阅《AWS 一般参考》中的区域和端点。	2017 年 11 月 29 日

更改	描述	日期
支持 ORC 格式 Amazon S3 清单文件	对于清单输出文件，除了逗号分隔值 (CSV) 文件格式之外，Amazon S3 现在还支持 Apache 优化行列式 (ORC) 格式。此外，现在您可以通过 Amazon Athena、Amazon Redshift Spectrum 以及其他工具（如 Presto、Apache Hive 和 Apache Spark）使用标准 SQL 语言查询 Amazon S3 清单。有关更多信息，请参阅 Amazon S3 清单。	2017 年 11 月 17 日
S3 存储桶的默认加密	Amazon S3 默认加密提供了一种方法来设置 S3 存储桶的默认加密行为。您可以对存储桶设置默认加密，以便在存储桶中存储所有对象时对这些对象进行加密。这些对象使用具有 Amazon S3 托管式密钥（SSE-S3）或 AWS 托管式密钥（SSE-KMS）的服务器端加密进行加密。有关更多信息，请参阅 为 Amazon S3 存储桶设置默认服务器端加密行为。	2017 年 11 月 06 日
Amazon S3 清单中的加密状态	Amazon S3 现在的支持包括 Amazon S3 清单中的加密状态，因此您查看对象在静态时如何加密，以用于合规性审计或其他用途。您还可以配置使用服务器端加密 (SSE) 或 SSE-KMS 来加密 Amazon S3 清单，从而相应地加密所有清单文件。有关更多信息，请参阅 Amazon S3 清单。	2017 年 11 月 06 日
跨区域复制 (CRR) 增强功能	跨区域复制现在支持以下功能： • 在跨账户方案中，您可以添加 CRR 配置，将副本所有权更改为拥有目标存储桶的 AWS 账户。有关更多信息，请参阅 更改副本拥有者。 • 默认情况下，Amazon S3 不复制源存储桶中使用具有 AWS KMS 中存储的密钥的服务器端加密创建的对象。在 CRR 配置中，您现在可以指示 Amazon S3 复制这些对象。有关更多信息，请参阅 复制加密对象（SSE-C、SSE-S3、SSE-KMS、DSSE-KMS）。	2017 年 11 月 06 日
欧洲（伦敦）区域	Amazon S3 现已在欧洲地区（伦敦）区域中提供。有关 Amazon S3 区域和端点的更多信息，请参阅《AWS 一般参考》中的区域和端点。	2016 年 12 月 13 日

更改	描述	日期
加拿大 (中部) 区域	Amazon S3 现已在加拿大（中部）区域中提供。有关 Amazon S3 区域和端点的更多信息，请参阅《AWS 一般参考》中的区域和端点。	2016 年 12 月 8 日
对象标签	Amazon S3 现在支持对象标签。利用对象标签，您可以对存储进行分类。对象键名前缀还可让您对存储进行分类，对象标签将向其添加另一个维度。 添加标签还可提供其他好处。包括： • 对象标签支持权限的精细访问控制 (例如，您可以向 IAM 用户授予对具有特定标签的只读对象的权限)。 • 指定生命周期配置时的精细控制。您可以指定标签以选择生命周期规则应用于的对象子集。 • 如果您已配置跨区域复制 (CRR)，则 Amazon S3 可以复制标签。您必须向为 Amazon S3 创建且要代入的 IAM 角色授予代表您复制对象的权限。 • 您还可以自定义 CloudWatch 指标和 CloudTrail 事件，以便按特定标签筛选条件显示信息。 有关更多信息，请参阅 使用标签对存储进行分类。	2016 年 11 月 29 日
Amazon S3 生命周期现在支持基于标签的筛选	Amazon S3 现在支持生命周期配置中基于标签的筛选。您现在可以指定生命周期规则，在该规则下，您可以指定键前缀、一个或多个对象标签或二者的组合以选择生命周期规则将应用于的对象子集。有关更多信息，请参阅 管理存储生命周期。	2016 年 11 月 29 日

更改	描述	日期
存储桶的 CloudWatch 请求指标	Amazon S3 现在支持在存储桶上发出的请求的 CloudWatch 指标。当您为存储桶启用这些指标时，指标每分钟报告一次。您还可以配置存储桶中的哪些对象将报告这些请求指标。有关更多信息，请参阅 使用 Amazon CloudWatch 监控指标。	2016 年 11 月 29 日
Amazon S3 清单	Amazon S3 现在支持存储清单。Amazon S3 清单每天或每周为 S3 存储桶或共享前缀（即，其名称以通用字符串开头的对象）提供对象及其对应元数据的平面文件输出。 有关更多信息，请参阅 Amazon S3 清单。	2016 年 11 月 29 日
Amazon S3 分析 - 存储类分析	新的 Amazon S3 分析 - 存储类分析特征可观察数据访问模式以帮助您确定何时将不常访问的 STANDARD 存储转换为 STANDARD_IA （IA 表示不频繁访问）存储类。在存储类分析在一段时间内观察到一组筛选出的数据的不常访问模式后，您可以使用分析结果来帮助改进您的生命周期配置。此特征还包括在指定的存储桶、前缀或标签级别对存储使用情况的详细日常分析，并且您可以将分析结果导出到 S3 存储桶。	2016 年 11 月 29 日
从 S3 Glacier 还原存档对象时新建加速和批量数据检索	在将存档对象还原至 S3 Glacier 时，Amazon S3 现在支持加速和批量数据检索以及标准检索。有关更多信息，请参阅恢复已归档的对象。	2016 年 11 月 21 日
CloudTrail 对象日志记录	CloudTrail 支持记录 Amazon S3 对象级别 API 操作，例如 GetObject 、PutObject 和 DeleteObject 。您可以将事件选择器配置为记录对象级别 API 操作。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon S3 API 调用。	2016 年 11 月 21 日
美国东部（俄亥俄）区域	Amazon S3 现已在美国东部（俄亥俄州）区域中提供。有关 Amazon S3 区域和端点的更多信息，请参阅《AWS 一般参考》中的区域和端点。	2016 年 10 月 17 日

更改	描述	日期
对 Amazon S3 Transfer Acceleration 的 IPv6 支持	Amazon S3 现在支持用于 Amazon S3 Transfer Acceleration 的 Internet 协议版本 6 (IPv6)。您可以使用全新的 Transfer Acceleration 双堆栈端点，通过 IPv6 连接到 Amazon S3。有关更多信息，请参阅 开始使用 Amazon S3 Transfer Acceleration。	2016 年 10 月 6 日
IPv6 支持	Amazon S3 现在支持 Internet 协议版本 6 (IPv6)。您可以使用双堆栈端点，通过 IPv6 访问 Amazon S3。有关更多信息，请参阅 通过 IPv6 向 Amazon S3 发出请求。	2016 年 8 月 11 日
亚太地区 (孟买) 区域	Amazon S3 现已在亚太地区（孟买）区域中提供。有关 Amazon S3 区域和端点的更多信息，请参阅《AWS 一般参考》中的区域和端点。	2016 年 6 月 27 日
Amazon S3 Transfer Acceleration	Amazon S3 Transfer Acceleration 可在您的客户端和 S3 存储桶之间实现快速、轻松、安全的远距离文件传输。Transfer Acceleration 利用 Amazon CloudFront 的全球分布式边缘站点。 有关更多信息，请参阅 使用 Amazon S3 Transfer Acceleration 配置快速、安全的文件传输。	2016 年 4 月 19 日
生命周期支持移除过期对象删除标记	生命周期配置 `Expiration` 操作现在允许您指示 Amazon S3 移除受版本控制的存储桶中的过期对象删除标记。有关更多信息，请参阅 用于描述生命周期操作的元素。	2016 年 3 月 16 日

更改	描述	日期
存储桶生命周期配置现在支持停止未完成的分段上传的操作	存储桶生命周期配置现在支持 AbortIncompleteMultipartUpload 操作，您可以使用该操作指示 Amazon S3 停止未在启动后的指定天数内完成的分段上传。当分段上传符合停止操作条件时，Amazon S3 将删除所有已上传的段并停止分段上传。 有关概念性信息，请参阅 Amazon S3 用户指南中的以下主题： • 中止分段上传 • 用于描述生命周期操作的元素 以下 API 操作已更新为支持新操作： • PUT 存储桶生命周期 – XML 配置现在允许您在生命周期配置规则中指定 AbortIncompleteMultipartUpload 操作。 • 列出段和启动分段上传 – 如果存储桶包含指定 AbortIncompleteMultipartUpload 操作的生命周期规则，则这两个 API 操作现在将返回两个额外的响应标头（x-amz-abort-date 和 x-amz-abort-rule-id ）。响应中的这些标头指示启动的分段上传何时符合停止操作的条件以及适用哪个生命周期规则。	2016 年 3 月 16 日
亚太区域（首尔）	Amazon S3 现已在亚太地区（首尔）区域中提供。有关 Amazon S3 区域和端点的更多信息，请参阅《AWS 一般参考》中的区域和端点。	2016 年 1 月 6 日

更改	描述	日期
新的条件键和分段上传变更	IAM 策略现在支持 Amazon S3 `s3:x-amz-storage-class` 条件键。有关更多信息，请参阅 使用条件键的存储桶策略示例。 您不再需要作为分段上传的发起者就能上传分段和完成上传。有关更多信息，请参阅 分段上传 API 和权限。	2015 年 12 月 14 日
重命名美国标准区域	区域名称字符串从“美国标准”变更为“美国东部（弗吉尼亚北部）”。此次只更新了区域名称，功能上没有任何变化。	2015 年 12 月 11 日
新存储类	Amazon S3 现在为存储对象提供了新存储类 STANDARD_IA（IA，适用于不频繁访问）。此存储类已针对长时间运行且不常访问的数据进行了优化。有关更多信息，请参阅 使用 Amazon S3 存储类。 生命周期配置特征的更新现在可允许您将对象转换为 STANDARD_IA 存储类。有关更多信息，请参阅 管理存储生命周期。 以前，跨区域复制特征对对象副本使用源对象的存储类。现在，当您配置跨区域复制时，您可以在目标存储桶中为创建的对象副本指定存储类。有关更多信息，请参阅 复制对象概述。	2015 年 9 月 16 日
AWS CloudTrail 集成	新的 AWS CloudTrail 集成能够记录您的 S3 存储桶中发生的 Amazon S3 API 活动。您可以使用 CloudTrail 来跟踪 S3 存储桶的创建或删除、访问控制修改或生命周期配置变更。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon S3 API 调用。	2015 年 9 月 1 日

更改	描述	日期
存储桶限额提升	Amazon S3 现在支持存储桶限额提升。默认情况下，客户可在其 AWS 账户中创建多达 100 个存储桶。需要更多存储桶的客户可通过提交服务限额提升来调高限制。有关如何提升存储桶限制的信息，请参阅《AWS 一般参考》中的 AWS 服务限额。有关更多信息，请参阅使用 AWS 软件开发工具包和存储桶配额、限制和局限性。	2015 年 8 月 4 日
一致性模型更新	Amazon S3 现在在美国东部（弗吉尼亚北部）区域对添加到 Amazon S3 的新对象支持先写后读一致性。在此更新之前，除美国东部（弗吉尼亚北部）区域之外的所有区域都对上传到 Amazon S3 的新对象支持先写后读一致性。利用此改进，Amazon S3 现在在所有区域对添加到 Amazon S3 的新对象都支持先写后读一致性。先写后读一致性允许您在 Amazon S3 中创建对象后立即检索对象。有关更多信息，请参阅 区域。	2015 年 8 月 4 日
事件通知	Amazon S3 事件通知进行了更新，以添加在删除对象时发送的通知并添加通过前缀和后缀匹配进行对象名称筛选的功能。有关更多信息，请参阅 Amazon S3 事件通知。	2015 年 7 月 28 日
Amazon CloudWatch 集成	借助新的 Amazon CloudWatch 集成，您可以通过适用于 Amazon S3 的 CloudWatch 指标监控和设置有关 Amazon S3 使用情况的警报。支持的指标包含 Standard 存储的总字节数、低冗余存储的总字节数以及给定 S3 存储桶的对象的总数。有关更多信息，请参阅 使用 Amazon CloudWatch 监控指标。	2015 年 7 月 28 日
支持删除和清空非空存储桶	Amazon S3 现在支持删除和清空非空存储桶。有关更多信息，请参阅 清空存储桶。	2015 年 7 月 16 日
Amazon VPC 端点的存储桶策略	Amazon S3 添加了对 Virtual Private Cloud（VPC）端点的存储桶策略的支持。您可以使用 S3 存储桶策略控制从特定 VPC 端点或特定 VPC 对存储桶的访问。VPC 端点易于配置，高度可靠，并且提供到 Amazon S3 的安全连接而无需网关或 NAT 实例。有关更多信息，请参阅 使用存储桶策略控制从 VPC 端点的访问。	2015 年 4 月 29 日

更改	描述	日期
事件通知	已更新 Amazon S3 事件通知，以支持切换到 AWS Lambda 函数的基于资源的权限。有关更多信息，请参阅 Amazon S3 事件通知。	2015 年 4 月 9 日
跨区域复制	Amazon S3 现在支持跨区域复制。跨区域复制是跨不同 AWS 区域 中的存储桶自动、异步地复制对象。有关更多信息，请参阅 复制对象概述。	2015 年 3 月 24 日
事件通知	Amazon S3 现在支持存储桶通知配置中的新事件类型和目标。在此版本之前，Amazon S3 仅支持 s3:ReducedRedundancyLostObject 事件类型和 Amazon SNS 主题作为目标。有关新事件类型的更多信息，请参阅 Amazon S3 事件通知。	2014 年 11 月 13 日
使用客户提供的加密密钥的服务器端加密	具有 AWS Key Management Service（AWS KMS）密钥的服务器端加密（SSE-KMS） Amazon S3 现已支持使用 AWS KMS 的服务器端加密。此特征允许您通过 AWS KMS 管理信封密钥，Amazon S3 调用 AWS KMS 以在您设置的权限范围内访问信封密钥。 有关使用 AWS KMS 的服务器端加密的更多信息，请参阅通过使用 AWS Key Management Service 的服务器端加密保护数据。	2014 年 11 月 12 日
欧洲（法兰克福）区域	Amazon S3 现已在欧洲（法兰克福）区域中提供。	2014 年 10 月 23 日

更改	描述	日期
使用客户提供的加密密钥的服务器端加密	Amazon S3 现在支持使用客户提供的加密密钥的服务器端加密 (SSE-C)。服务器端加密使您能够请求 Amazon S3 加密您的静态数据。在使用 SSE-C 时，Amazon S3 将使用您提供的自定义加密密钥加密您的对象。由于 Amazon S3 会自动执行加密，因此您可以使用自己的加密密钥，而无需编写或执行自己的加密代码。 有关 SSE-C 的更多信息，请参阅服务器端加密（使用客户提供的加密密钥）。	2014 年 6 月 12 日
版本控制的生命周期支持	在此版本之前，仅对不受版本控制的存储桶支持生命周期配置。现在您对不受版本控制和启用了版本控制的存储桶都可以配置生命周期。有关更多信息，请参阅 管理存储生命周期。	2014 年 5 月 20 日
修订了访问控制主题	修订了 Amazon S3 访问控制文档。有关更多信息，请参阅 Amazon S3 的身份和访问管理。	2014 年 4 月 15 日
修订了服务器访问日志记录主题	修订了服务器访问日志记录文档。有关更多信息，请参阅 使用服务器访问日志记录来记录请求。	2013 年 11 月 26 日
.NET SDK 示例已更新为版本 2.0	本指南中的 .NET SDK 示例现在符合版本 2.0 标准。	2013 年 11 月 26 日
HTTP 上的 SOAP 支持已弃用	HTTP 上的 SOAP 支持已弃用，但是仍可在 HTTPS 上使用。不支持将新 Amazon S3 特征用于 SOAP。我们建议您使用 REST API 或 AWS SDK。	2013 年 9 月 20 日

更改	描述	日期
IAM 策略变量支持	IAM 策略语言现在支持变量。在评估策略时，任何策略变量都会替换为来自经身份验证的用户会话的上下文相关信息提供的值。您可以使用策略变量来定义通用策略，无需显式列出策略的所有部分。有关策略变量的更多信息，请参阅《IAM 用户指南》中的 IAM 策略变量概述。 有关 Amazon S3 中策略变量的示例，请参阅Amazon S3 基于身份的策略示例。	2013 年 4 月 3 日
对申请方付款的控制台支持	现在，您可以使用 Amazon S3 控制台为申请方付款配置存储段。有关更多信息，请参阅 使用申请方付款存储桶进行存储传输和使用。	2012 年 12 月 31 日
对网站托管的根域支持	Amazon S3 现已支持在根域中托管静态网站。网站访问者在其浏览器中访问网站时，无需在 Web 地址中指定 www（例如，他们可以使用 example.com，而不是 www.example.com）。许多客户已在 Amazon S3 上托管了静态网站，访问者可通过 www 子域（例如 www.example.com）进行访问。此前，要支持根域访问，您必须在运行自己的 Web 服务器时将根域请求从浏览器以代理的方式转至您在 Amazon S3 上的网站。运行 Web 服务器对请求进行代理会导致出现额外成本、运转负荷和其他潜在的故障点。现在，对于 www 和根域地址，您都可以充分利用 Amazon S3 的高可用性和耐用性了。有关更多信息，请参阅 使用 Amazon S3 托管静态网站。	2012 年 12 月 27 日
控制台修改	对 Amazon S3 控制台进行了更新。并对与控制台相关的文档主题做出了相应修改。	2012 年 12 月 14 日

更改	描述	日期
支持将数据存档到 S3 Glacier	Amazon S3 现已支持的存储选项可让您使用 S3 Glacier 的低成本存储服务来归档数据。要将对象归档，您需要定义归档规则来确定对象以及您希望 Amazon S3 据之将这些对象归档到 S3 Glacier 的时间范围。通过使用 Amazon S3 控制台，或以编程方式使用 Amazon S3 API 或 AWS SDK，您可以轻松地在存储桶上设置规则。 有关更多信息，请参阅 管理存储生命周期。	2012 年 11 月 13 日
对网页重定向的支持	对于配置为网站的存储段，Amazon S3 现已支持将对象的请求重定向到相同存储段中的另一个对象或外部 URL。有关更多信息，请参阅 （可选）配置网页重定向。 有关托管网站的信息，请参阅使用 Amazon S3 托管静态网站。	2012 年 10 月 4 日
对跨域资源共享 (CORS) 的支持	Amazon S3 现已支持跨域资源共享 (CORS) 功能。一个域中加载的客户端 Web 应用程序可借助 CORS 定义的方式与不同域中的资源进行交互，或访问这些资源。借助 Amazon S3 中的 CORS 支持，您可以基于 Amazon S3 构建各种富客户端 Web 应用程序，并选择性地允许跨域访问您的 Amazon S3 资源。有关更多信息，请参阅 使用跨源资源共享 (CORS)。	2012 年 8 月 31 日
对成本分配标记的支持	Amazon S3 现已支持成本分配标记，这允许您为 S3 存储桶加标签，从而可以更轻松地根据项目或其他条件跟踪其成本。有关将标记用于存储桶的更多信息，请参阅 使用成本分配 S3 存储桶标签。	2012 年 8 月 21 日

更改	描述	日期
对存储桶策略中受 MFA 保护的 API 访问的支持	Amazon S3 现已支持受 MFA 保护的 API 访问，这项特征在访问 Amazon S3 资源时可强制执行 AWS Multi-Factor Authentication 以提供更高级别的安全性。它是一项安全特征，要求用户通过提供有效 MFA 代码来证明实际拥有 MFA 设备。有关更多信息，请参阅 AWS Multi-Factor Authentication。现在，您可以对任何访问 Amazon S3 资源的请求要求进行 MFA 身份验证。 为了强制执行 MFA 身份验证，Amazon S3 现已支持存储桶策略中的 `aws:MultiFactorAuthAge` 密钥。有关存储桶策略示例，请参阅 需要 MFA。	2012 年 7 月 10 日
对象到期支持	借助“对象到期”功能，您可以设定在已配置时间段后自动删除数据。通过向存储桶添加生命周期配置，可以设置对象到期规则。	2011 年 12 月 27 日
增加了新的支持区域	Amazon S3 现已支持南美洲（圣保罗）区域。有关更多信息，请参阅 访问和列出 Amazon S3 存储桶。	2011 年 12 月 14 日
多对象删除	Amazon S3 现已支持多对象删除 API，使您能够删除单个请求中的多个对象。借助此特征，您可以比使用多个单独的 DELETE 请求更快地从 Amazon S3 删除大量对象。有关更多信息，请参阅 删除 Amazon S3 对象。	2011 年 12 月 7 日
增加了新的支持区域	Amazon S3 现已支持美国西部（俄勒冈）地区。有关更多信息，请参阅存储桶和区域。	2011 年 11 月 8 日
文档更新	文档错误修复。	2011 年 11 月 8 日
文档更新	除了文档错误修复，本版本还包括以下增强功能： • 使用 AWS SDK for PHP 和 AWS SDK for Ruby（请参阅指定具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3））的新服务器端加密部分。	2011 年 10 月 17 日

更改	描述	日期
服务器端加密支持	Amazon S3 现已支持服务器端加密。它使您能够请求 Amazon S3 静态加密数据，即，当 Amazon S3 将数据写入其数据中心内的磁盘时，加密您的对象数据。除了 REST API 更新，AWS SDK for Java 和 .NET 可提供必要的功能来请求服务器端加密。当使用 AWS Management Console 上传对象时，还可以请求服务器端加密。要了解有关数据加密的更多信息，请转至使用数据加密。	2011 年 10 月 4 日
文档更新	除了文档错误修复，本版本还包括以下增强功能： • 在 提出请求 部分中添加了 Ruby 和 PHP 示例。 • 添加了介绍如何生成和使用预签名 URL 的部分。有关更多信息，请参阅使用预签名 URL 共享对象 和使用预签名 URL 共享对象。 • 更新了介绍 AWS Explorer for Eclipse 和 Visual Studio 的部分。有关更多信息，请参阅 使用 AWS SDK 通过 Amazon S3 进行开发。	2011 年 9 月 22 日
对使用临时安全凭证发送请求的支持	除了使用您的 AWS 账户和 IAM 用户安全凭证将经身份验证的请求发送到 Amazon S3，您现在还可以使用从 AWS Identity and Access Management（IAM）获取的临时安全凭证发送请求。您可以使用 AWS Security Token Service API 或 AWS SDK 包装程序库从 IAM 请求这些临时凭证。您可以请求这些临时安全凭证以供自己使用，也可以提供给联合身份用户和应用程序使用。此特征使您能够管理AWS之外的用户，并为他们提供临时安全凭证来访问AWS资源。 有关更多信息，请参阅 提出请求。 有关 IAM 对临时安全凭证的支持的更多信息，请参阅《IAM 用户指南》中的临时安全凭证。	2011 年 8 月 3 日

更改	描述	日期
扩展了分段上传 API 以支持复制多达 5 TB 的对象	在此版本之前，Amazon S3 API 支持复制的对象最大为 5 GB。为了支持复制大于 5 GB 的对象，Amazon S3 现已通过新的操作 `Upload Part (Copy)` 扩展了分段上传 API。您可以使用该分段上传操作来复制多达 5 TB 的对象。有关更多信息，请参阅 复制、移动和重命名对象。 有关分段上传 API 的概念性信息，请参阅使用分段上传来上传和复制对象。	2011 年 21 月 6 日
已禁用通过 HTTP 的 SOAP API 调用	为了提高安全性，已禁用通过 HTTP 的 SOAP API 调用。经身份验证的和匿名的 SOAP 请求必须使用 SSL 发送到 Amazon S3。	2011 年 6 月 6 日
IAM 支持跨账户的委派	以前，要访问 Amazon S3 资源，IAM 用户需要从父 AWS 账户和 Amazon S3 资源拥有者这两者处获得许可。借助跨账户访问，IAM 用户现在仅需要从所有者账户获得许可。也即，如果资源拥有者向 AWS 账户授予访问权限，则 AWS 账户现在可以向其 IAM 用户授予对这些资源的访问权限。 有关更多信息，请参阅《IAM 用户指南》中的创建向 IAM 用户委派权限的角色。 有关在存储桶策略中指定主体的更多信息，请参阅 存储桶策略的主体。	2011 年 6 月 6 日
新链接	此服务的端点信息现在位于《AWS 一般参考》中。有关更多信息，请参阅《AWS 一般参考》https://docs.aws.amazon.com/general/latest/gr/rande.html中的“区域和端点”。	2011 年 3 月 1 日

更改	描述	日期
支持在 Amazon S3 中托管静态网站	Amazon S3 增强了对托管静态网站的支持。其中包括支持索引文档和自定义错误文档。使用这些特征时，访问存储桶的根目录或子文件夹（例如 `http://mywebsite.com/subfolder`）的请求会返回索引文档，而不是存储桶中对象的列表。遇到错误时，Amazon S3 会返回用户自定义的错误消息，而不是 Amazon S3 错误消息。有关更多信息，请参阅 使用 Amazon S3 托管静态网站。	2011 年 6 月 6 日
此服务的端点信息现在位于《AWS 一般参考》中。有关更多信息，请参阅《AWS 一般参考》https://docs.aws.amazon.com/general/latest/gr/rande.html中的“区域和端点”。	2011 年 3 月 1 日	
支持在 Amazon S3 中托管静态网站	Amazon S3 增强了对托管静态网站的支持。其中包括支持索引文档和自定义错误文档。使用这些特征时，访问存储桶的根目录或子文件夹（例如 `http://mywebsite.com/subfolder`）的请求会返回索引文档，而不是存储桶中对象的列表。遇到错误时，Amazon S3 会返回用户自定义的错误消息，而不是 Amazon S3 错误消息。有关更多信息，请参阅 使用 Amazon S3 托管静态网站。	2011 年 2 月 17 日
支持响应标头 API	GET Object REST API 现在允许您为每个请求更改 REST GET Object 请求的响应标头。即，您可以改变响应中的对象元数据，无需改变对象本身。有关更多信息，请参阅 下载对象。	2011 年 1 月 14 日

更改	描述	日期
支持大型对象	Amazon S3 已将 S3 存储桶中可存储对象的最大大小从 5 GB 提高到 5 TB。如果您使用 REST API，则可以在单个 PUT 操作中上传最多 5GB 的对象。对于大型对象，必须使用分段上传 REST API 来分段上传对象。有关更多信息，请参阅 使用分段上传来上传和复制对象。	2010 年 12 月 9 日
分段上传	分段上传可以更快、更灵活地上传到 Amazon S3。它允许您上传单个对象作为一组分段。有关更多信息，请参阅 使用分段上传来上传和复制对象。	2010 年 11 月 10 日
支持存储桶策略中的规范 ID	现在您可以在存储桶策略中指定规范 ID。有关更多信息，请参阅 存储桶策略的主体	2010 年 9 月 17 日
Amazon S3 与 IAM 结合使用	该服务现已集成 AWS Identity and Access Management（IAM）。有关更多信息，请转到《IAM 用户指南》中的与 IAM 结合使用的 AWS 服务。	2010 年 9 月 2 日
通知	Amazon S3 通知特征使您能够配置存储桶，以便当 Amazon S3 在存储桶上检测到关键事件时，Amazon S3 可向 Amazon Simple Notification Service（Amazon SNS）主题发布消息。有关更多信息，请参阅设置存储桶事件的通知。	2010 年 7 月 14 日
存储桶策略	存储桶策略是用来跨存储桶、对象和对象集设置访问权限的访问管理系统。此功能可补充（在许多情况下，可替代）访问控制列表。有关更多信息，请参阅 Amazon S3 的存储桶策略。	2010 年 7 月 6 日
在所有区域中可用的路径类型语法	在美国传统区域，或者如果存储桶与请求的端点位于相同区域，Amazon S3 现在支持任何存储桶的路径类型语法。有关更多信息，请参阅虚拟托管。	2010 年 6 月 9 日
欧洲（爱尔兰）的新端点	Amazon S3 现在为欧洲（爱尔兰）提供端点：`http://s3-eu-west-1.amazonaws.com` 。	2010 年 6 月 9 日

更改	描述	日期
控制台	您现在可以通过AWS Management Console使用 Amazon S3。您可以在《Amazon Simple Storage Service 用户指南》中阅读有关控制台中所有 Amazon S3 功能的信息。	2010 年 6 月 9 日
去冗余	Amazon S3 现在使您能够通过在去冗余的 Amazon S3 中存储对象，来降低存储成本。有关更多信息，请参阅低冗余存储。	2010 年 5 月 12 日
增加了新的支持区域	Amazon S3 现已支持亚太地区（新加坡）区域。有关更多信息，请参阅存储桶和区域。	2010 年 4 月 28 日
对象版本控制	本版本引入了对象版本控制。所有对象现在都有键和版本。如果您启用对存储段的版本控制，则 Amazon S3 将为所有添加到存储段的对象提供唯一的版本 ID。此特征使您能够从意外覆盖和删除中恢复。有关更多信息，请参阅版本控制和使用版本控制。	2010 年 2 月 8 日
增加了新的支持区域	Amazon S3 现在支持美国西部（北加利福尼亚）区域。对此区域的请求的新端点为 `s3-us-west-1.amazonaws.com` 。有关更多信息，请参阅存储桶和区域。	2009 年 12 月 2 日
AWS SDK for .NET	对于更喜欢使用 .NET 语言特定的 API 操作而不是使用 REST 或 SOAP 来构建应用程序的软件开发人员，AWS 现在为他们提供了库、示例代码、教程和其他资源。这些库提供了一些基本功能 (未包括在 REST 或 SOAP API 中)，比如请求身份验证、请求重试和错误处置，以便您轻松地开始工作。有关特定语言的库和资源的更多信息，请参阅使用 AWS SDK 通过 Amazon S3 进行开发。	2009 年 11 月 11 日

AWS 术语表

有关最新的 AWS 术语，请参阅《AWS 词汇表参考》中的 AWS 词汇表。